Departamento de Computación
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Herramienta automática para la segmentación de la arteria aorta a partir de imágenes de tomografía computarizada

Estudante: André García Gómez
Director/a/es/as: María Noelia Barreira Rodríguez
Manuel Francisco González Penedo

A Coruña, 31 de agosto de 2019.

Dedicado a mis padres, Elena Gómez Gálkina y Pablo García Tahoces, que hicieron posible que yo consiguiera llegar hasta aquí.

Agradecementos

Especial agradecimientos a mi tutora María Noelia Barreira Rodríguez por su gran paciencia y ayuda a la hora de realizar el sistema como a la corrección de la memoria. Agradecimiento también a Manuel Francisco González Penedo por aprobarlo y permitir la realización del proyecto como trabajo de final de carrera y por último, agradecer a mi padre Pablo García Tahoces por prestarme su tiempo y sus años de experiencia para ayudarme en el proceso.

Resumen

La aorta es la principal arteria del cuerpo humano. Sale del ventrículo izquierdo del corazón y da origen a todas las arterias del sistema circulatorio con excepción de las arterias pulmonares, siendo así uno de los principales canales de transmisión de oxígeno del cuerpo. Su cuidado es fundamental dado que una mala salud de la misma pueden tener un nivel alto de mortalidad para el propio paciente. Gran parte de las enfermedades de la aorta pueden detectarse por su forma, como es el caso del aneurisma, enfermedad que va expandiendo las paredes de la aorta pudiendo llegar a romperla. También a partir de la forma se pueden detectar malformaciones naturales de la misma que pueden dificultar la vida del paciente.

Actualmente existen sistemas de segmentación automática de la aorta con metodologías tradicionales de procesado de imagen como es el caso del crecimiento por regiones o el seguimiento de la forma de la aorta. Sin embargo, son lentos y, aunque generan buenos resultados en la gran parte de los casos, tienen dificultades a la hora de segmentar aortas con formas poco comunes. Por otra parte, los sistemas de *deep learning* están siendo utilizados en los últimos años de forma masiva para resolver problemas en el ámbito de la imagen médica. Esto es debido a su precisión y rapidez en el procesado gracias a la aparición de nuevas arquitecturas de redes neuronales y el desarrollo de hardware especializado en este tipo de tecnologías.

En este proyecto se desarrollará una metodología para la segmentación automática de la forma de la aorta a partir de imágenes de tomografía computarizada (CT) en 3D. Dicho sistema estará basado en técnicas de *deep learning* y visión artificial. En concreto, se probará la arquitectura de red *UNet* para abordar la segmentación preliminar de las estructuras vasculares presentes en la imagen y se desarrollarán algoritmos de procesado de imagen para realizar la reconstrucción completa de la arteria. La metodología desarrollada se incluirá en una herramienta de visualización que también se encargará de realizar el proceso de segmentación

Palabras clave:

- Aprendizaje automático
- Aorta
- UNet
- Segmentación
- Redes neuronales

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se hará una breve introducción sobre el proyecto haciendo mención a los objetivos del sistema. También se describirá la organización de la memoria y antecedentes del mismo.

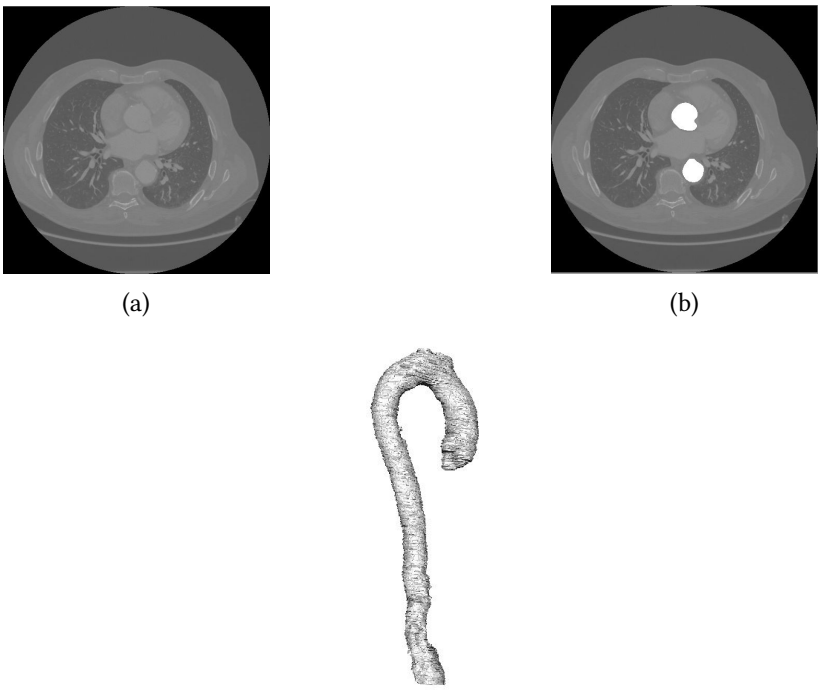
1.1 Motivación

La aorta es la principal arteria del cuerpo humano, sale del ventrículo izquierdo del corazón y da origen a todas las arterias del sistema circulatorio con excepción de las arterias pulmonares, siendo así uno de los principales canales de transmisión de oxígeno del cuerpo (Apendice A). Su cuidado es fundamental dado que una mala salud de la misma pueden tener un nivel alto de mortalidad para el propio paciente. Gran parte de las enfermedades de la aorta pueden detectarse por su forma, como es el caso del aneurisma, enfermedad que va dilatando las paredes de la aorta pudiendo llegar a romperla. También a partir de la forma se pueden detectar malformaciones naturales de la misma que pueden dificultar la vida del paciente.

Hoy en día para examinar la aorta se usan imágenes de tomografía computarizada (CT). Sin embargo su observación es complicada dado a la cantidad de elementos del organismo que la rodean lo cual dificulta el proceso (Figura 1.1). El uso de herramientas automáticas para la segmentación de la misma facilita su observación.

1.2 Estado del arte

La segmentación de imágenes es la tarea de agrupar y separar los elementos de una imagen con el fin de facilitar su clasificación u observación de la misma. En imagen médica estas técnicas se utilizan para separar distintas partes del organismo con el fin de poder analizarlas y encontrar defectos que puedan afectar la salud del paciente. Dicha segmentación originalmen-



(a) (b)

(c)

Figura 1.1: **Tomografía computorizada de un paciente**. a) Imagen original. b) Imagen original con la aorta marcada. c) Segmentación 3D del paciente

te se hacía de manera manual. Sin embargo, con la llegada de las técnicas de procesamiento de imagen en informática se intenta crear métodos automáticos. Estos métodos presentan dificultades dado al bajo contraste y separación que hay entre las distintas partes de nuestro organismo.

Métodos tradicionales para la segmentación automática en imagen médica fueron diseñados a partir del conocimiento de expertos sobre el campo, intentando replicar la manera en la que ellos realizan la segmentación. Estos métodos aprovecharon algoritmos genéricos en procesamiento de imagen como el seguimiento de la forma, detección de esquinas y propagación de puntos, entre otros [3].

Con la llegada del aprendizaje automático han aparecido nuevas técnicas de segmentación. Cabe destacar que aquellas basadas en el uso de redes neuronales que utilizan grandes bases de datos para su entrenamiento. De esta forma, se han mejorando los resultados obtenidos previamente. Sin embargo, estos métodos tienen limitaciones a la hora debido a la dificultad de encontrar grandes bases de datos marcadas por el experto que permitan poder entrenar las redes neuronales. Debido a todo ello, se han desarrollado nuevas arquitecturas especializadas para su uso en imagen médica. Así, la UNet [4] ha conseguido resultados bastante importantes desde su introducción. Su principal característica es que han sido diseñada para obtener buenos resultados con bases de datos de tamaño más resucido que las empleadas en otros campos, como puede ser la clasificación de objetos genéricos, tales como *ImageNet* [5], donde se dispone de millones de casos para realizar el entrenamiento.

Para la segmentación de la aorta, han sido también desarrolladas diferentes técnicas de segmentación automática. Así, Zheng et al. utilizaron una UNet para la segmentación de la aorta abdominal [6]. López-Linares et al. utilizaron una variante de la Holistically-nested Edge Detection (HED) para la segmentación de trombos en aorta abdominal [7]. Finalmente, Tahoces et al. han desarrollado una técnica de segmentación de la aorta torácica basado en el uso de contornos activos paramétricos [3] a partir de una algoritmo de seguimiento de elipses [8].

1.3 Objetivos

Este trabajo tiene dos objetivos. El principal objetivo es crear un método que sea capaz de segmentar la aorta de manera rápida y con buena precisión, utilizando métodos de aprendizaje automático basados en redes neuronales y procesado de imágenes. El sistema partirá de imágenes CT en 3D de un paciente y devolverá una representación 3D de la aorta. El se-

gundo objetivo de este trabajo es el diseño y codificación de una aplicación de escritorio que permita usar el método de segmentación y visualizar sus resultados facilitando el uso de la herramienta.

1.4 Organización de la memoria

La memoria se divide en distintos capítulos que se ocupan de explicar las partes del proyecto:

- **En el capítulo 1** se ha realizado una breve introducción al trabajo.
- **En el capítulo 2**: se explican las distintas definiciones y conceptos necesarios para el entendimiento del proyecto.
- **En el capítulo 3** se listan todas las librerías, recursos y entornos de trabajo utilizados a lo largo del proyecto.
- **En el capítulo 4** se muestra la gestión que se ha llevado a cabo.
- **En el capítulo 5** se explican todos los métodos que se desarrollaron para cada una de las partes del proyecto.
- **En el capítulo 6** se muestran los resultados obtenidos en el proyecto.
- **En el capítulo 7** se explica el desarrollo de la aplicación.
- Finalmente, **en el capitulo 8** se presentan las conclusiones finales y se describen los posibles trabajos futuros.

Capítulo 2

Fundamentos teóricos

En este capítulo se abordarán los fundamentos teóricos necesarios para la correcta comprensión de este proyecto. Se realizará una breve introducción de las redes neuronales, explicando su funcionamiento y origen. También se detallarán el tipo de redes utilizadas en el desarrollo del proyecto así como las métricas utilizadas para validar su correcto funcionamiento.

2.1 Introducción

Las redes neuronales fueron creadas en 1943 como método para la resolución de problemas inspirándose en el funcionamiento del cerebro humano. Sin embargo, no obtuvieron relevancia hasta la actualidad dado la escasez de datos para entrenarlas y la escasa capacidad computacional disponible por entonces.

Una de las arquitecturas más básicas es el perceptrón. Está formado por un conjunto de parámetros llamados pesos w y bias b que deben ser entrenados en el proceso gracias al algoritmo de *backpropagation* introducido en los años 70 [9]. La idea principal de dicho algoritmo es una sucesión de iteraciones con el objetivo de calibrar el valor de los pesos para obtener los resultados deseados. Esto es solo posible si tenemos grandes bases de datos con que entrenarlas. Cada neurona tiene una estructura lineal donde x_i son los datos de entrada e Y los de salida. Un modelo de redes neuronales sería un conjunto de capas formadas por neuronas conectadas entre sí (Figura 2.1).

Inicialmente los resultados de las redes perceptrón consistían en dos valores (0,1) calculados a partir de la siguiente expresión:

$$Y = \begin{cases} 1 & Si \quad w \cdot x + b > 0, \\ 0 & resto \end{cases} \tag{2.1}$$

donde w representa los pesos, x los valores de entrada y b las bias. Sin embargo, se fueron

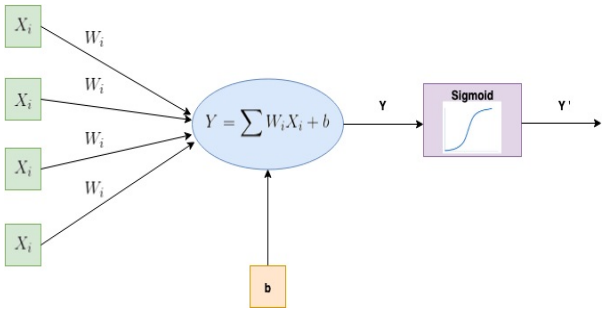


Figura 2.1: **Arquitectura perceptrón**

creando otras funciones llamadas funciones de activación que devuelven un rango más amplio de valores. La función *sigmoid* (Figura 2.1) es una de ellas. Esta función normaliza el valor de salida de tal manera que esté en un rango de valores entre [0,1].

Las redes neuronales clásicas presentaban un gran avance para problemas de clasificación o con gran número de variables con respecto a los métodos tradicionales. Sin embargo eran muy caras computacionalmente y poco eficientes para resolver problemas con número reducido de variables respecto a otros métodos.

2.2 Redes Neuronales Convolucionales (CNN)

Las CNN fueron creadas para facilitar el tratamiento de imágenes en las redes neuronales. Fueron propuestas en los años 80 y estaban inspiradas en los campos receptivos de la corteza visual de orgasnimos vivos. Sin embargo, no fue hasta la llegada del nuevo milenio con la aparición de las muevas implementaciones en GPU que estos modelos llegaron a ser eficientes, debido a a su elevado coste computacional [10]. Anteriormente para clasificar una imagen con arquitectura de redes neuronales tradicionales, se necesitaba un número alto de paramétros y un mayor tiempo de entrenamiento. Esto se puede ver con facilidad a la hora de entrenar un modelo que sea capaz de clasificar imágenes de números escritos a mano. Mientras que para obtener una buena exactitud a la hora de clasificar con un modelo completamente formado por perceptrón, un 94 por ciento de acierto, se necesitan un total de 407.050 parámetros y 30 etapas de entrenamientos (*epochs*) para imágenes de un tamaño 28 x 28. En un modelo mixto de CNN y perceptrón con un total de 140.682 parámetros y 4 *epochs* se consigue una exactitud del 97 por ciento [9].

El funcionamiento de las CNN parte de tres ideas básicas: campos receptivos locales, pesos compartidos y capas de agrupación:

- **Campos receptivos locales**: En la arquitectura perceptrón se analiza cada imagen píxel a píxel. En las CNN se analizan conjuntos de píxeles. Esto se hace a través de

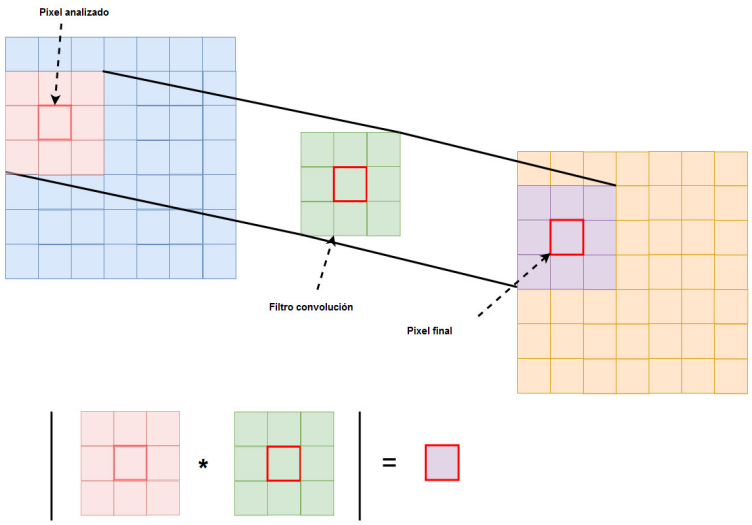


Figura 2.2: **Capa convolución**: asigna a cada pixel de la imagen el resultado del producto de la matriz formada por el propio pixel y sus vecinos por el filtro de convolución.

$kernels$ que van obteniendo los valores de un punto determinado y los vecinos que lo rodean [9] (Figura 2.2) utilizando la operación de convolución.

- **Pesos compartidos**: A diferencia de la arquitectura perceptrón en la que cada píxel tiene asignado un peso determinado, en las CNN todas las neuronas de una capa comparten los mismos pesos, los cuales operan en distintas partes de la imagen.
- **Capas de *pooling***: Las CNN también tienen capas encargadas de simplificar la información obtenida agrupando regiones en un mismo punto. La fórmula más popular es con operaciones de *Max pooling* (Figura 2.3) donde una región se representa con solo su valor máximo.

La función de activación más utilizada en las CNN es la rectificación lineal (ReLU) por su efectividad y simplicidad computacional, dando en general muy buenos resultados en problemas de procesamiento de imágenes tanto en problemas de segmentación como de clasificación. Su ecuación representativa es la siguiente:

$$f(x) = max(0, x) \tag{2.2}$$

2.3 UNet

La red UNet es una arquitectura de aprendizaje automático creada en la universidad de Freiburg especializada en la segmentación de imagen médica [4]. No necesita grandes bases

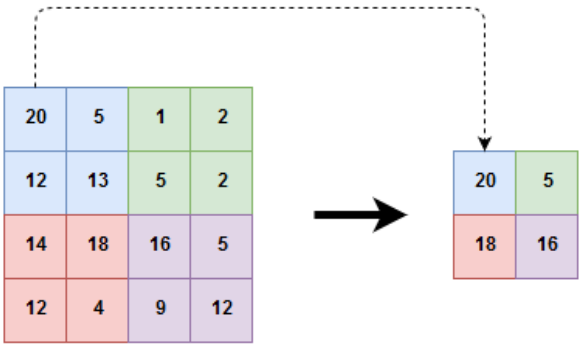


Figura 2.3: **Max Pooling**: selecciona los valores máximos de las distintas regiones que forman la matriz original.

de datos para ser entrenada y obtiene buenas segmentaciones basadas en la forma y el color de las distintas partes de la imagen. El modelo consta de dos partes (Figura 2.4). La primera es la parte de bajada que se ocupa de reducir la imagen en sus características generales. Consiste en la repetición de la aplicación de dos capas 3x3 convolucionales, cada una con una ReLU y una capa *maxpooling* 2x2. La segunda es la parte de subida que se ocupa de generar la imagen segmentada. Es similar a la primera dado que dispone de las mismas capas convolucionales pero acompañadas de una capa *upconvolution* encargada de expandir la imagen. Las capas *upconvolution* se ocupan de expandir el tamaño del resultado de la capa anterior a partir del uso de interpolaciones, mecanismo usado en la UNet para devolver la imagen a su tamaño original.

De esta forma la primera parte se ocupa de reducir la imagen en sus componentes principales mientras que la segunda en generar la imagen con la segmentación realizada. Para mantener la calidad de la imagen hay operaciones de concatenación entre ambas partes para que se mantengan las formas en su forma original.

2.4 Algoritmos de optimización

Estos algoritmos se utilizan para la optimización del proceso de aprendizaje en los modelos de *Deep Learning*. Su funcionamiento se puede entender al observar cómo funciona el algoritmo descenso de gradiente estocástico (SGD) también conocido como descenso de gradiente incremental [11], en el que trata de alcanzar el coste global mínimo en un problema determinado a partir de una serie de iteraciones.

Por cada iteración se debe reducir el gradiente actualizando los parámetros del problema para acercarse así más a la solución óptima. Esto se calcula gracias a la función de pérdida que devuelve el coste del problema, valor utilizado para acercarnos a dicha solución. Cabe

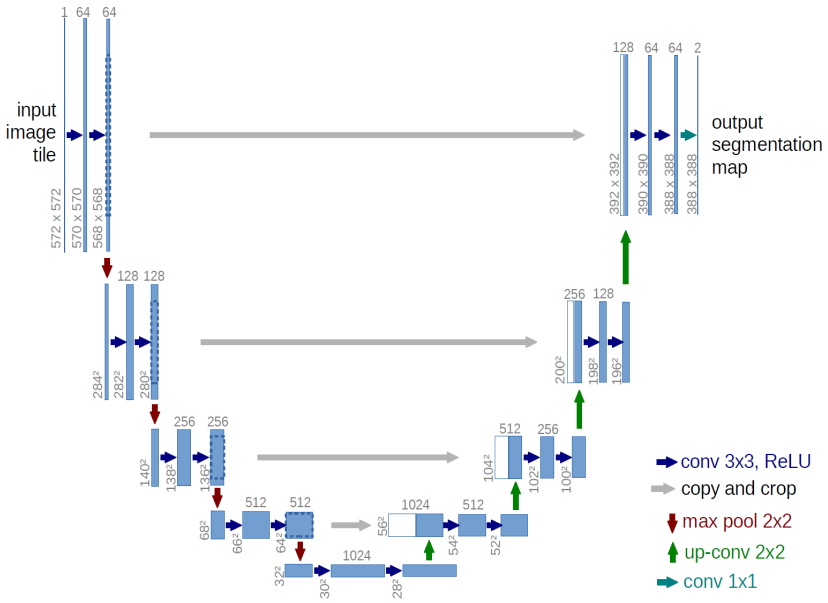


Figura 2.4: **Arquitectura UNet**

destacar que a pesar de que su intención es alcanzar el mínimo global puede terminar en un mínimo local.

Una alternativa al SGD es el algoritmo *Adam* [12]. Hoy en día es uno de los métodos de optimización de aprendizaje más populares por sus múltiples características entre las que cabe destacar: su sencillez a la hora de ser implementado, su eficiencia computacional y sus reducidos requisitos de memoria. Está implementado por defecto en las principales librerías actuales como TensorFlow, Keras o Torch.

Adam es un método de optimización de aprendizaje adaptativo, esto quiere decir que va modificando su ratio de aprendizaje automáticamente a lo largo del proceso. El nombre *Adam* deriva del término momento adaptativo. En el contexto se puede ver como una combinación de *RMSProp* y *momento* [10]. El momento acumula una decadencia exponencial promedio móvil de los anteriores gradientes y continúa moviéndose en esa dirección, *RMSProp* es otro algoritmo de aprendizaje adaptativo creado en 2012 que a su vez es una modificación de *AdaGrad* [10].

2.5 Métricas

Una de las partes más importantes de los modelos de aprendizaje automático es conocer cómo de bien han aprendido y qué porcentaje de acierto tienen. Para ello hay distintos métodos posibles dependiendo cual sea el problema a resolver.

A la hora de entrenar la red también es necesario disponer de un valor que nos indique

cómo de desviados estamos del resultado óptimo. Este valor viene dado por la función de pérdida (*loss*).

Una de las funciones de pérdida más populares es *Categorical Cross Entropy*, utilizada para la comparación de la distribución de una predicción. Solo usa una fila de categorización por lo que se debe usar en casos de que solo un resultado puede ser correcto. Otras funciones son el coeficiente Sørensen–Dice y el coeficiente de Jaccard.

2.5.1 Coeficiente Sørensen–Dice (DSC)

Se utiliza para calcular la diferencia entre los datos de entrada X y los datos de salida Y del modelo [13], siendo su rango de valores 0 si la similitud es nula y 1 si son idénticos:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.3}$$

En aprendizaje automático se usa especialmente para saber la exactitud obtenida en problemas de segmentación de imagen. Popularmente se usa como funciones de pérdida -DSC o 1 - DSC. Sin embargo, al disponer de una base de datos desbalanceada el índice Jaccard da mejores resultados.

2.5.2 Coeficiente de Jaccard

También conocido como intersección sobre la unión [13], se usa para encontrar la similitud y diversidad de un conjunto de datos:

$$\begin{aligned} Jaccard &= \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \\ &0 <= J(X,Y) >= 1 \end{aligned} \tag{2.4}$$

$J(X,Y) \in [0.7, 0.99]$ se consideran buenos resultados. Si $J(X,Y) = 1$ se considera que X e Y están vacíos.

Capítulo 3

Análisis de tecnologías y herramientas

En este capítulo se mencionarán las distintas tecnologías y herramientas utilizadas durante el proyecto. También, se explicará la base de datos y los distintos entornos de trabajos utilizados.

3.1 Tecnologías de desarrollo

Se describirán las tecnologías necesarias para la creación y desarrollo del proyecto. Estás tecnologías están principalmente relacionadas con aprendizaje automático, procesado de imágenes y optimización de los distintos procesos realizados a lo largo del proyecto.

3.1.1 Lenguajes de programación

Se utilizará como lenguaje de programación Python 3.7, un lenguaje interpretado, de alto nivel y de propósito general. Creado por Guido van Rossum y lanzado por primera vez en 1991, la filosofía de diseño de Python enfatiza la legibilidad del código con su uso de espacios en blanco significativos. Su sintaxis y enfoque orientado a objetos tiene como objetivo ayudar a los programadores a escribir código claro y lógico para proyectos pequeños y grandes [14].

3.1.2 Librerías

La lista de librerías de ingeniería de datos utilizadas a lo largo de la fase de desarrollo es la siguiente:

- **TensorFlow**: Librería de código libre para la investigación y producción de sistemas con aprendizaje automático. Fue desarrollado por Google Brain para uso interno de Google, siendo revelado al público en noviembre de 2015 [15].

- **Keras**: Librería de código libre desarrollada en python capaz de ejecutarse encima de TensorFlow, Theano y Microsoft Cognitive Toolkit. Fue diseñada para facilitar la experimentación de sistemas de aprendizaje automático haciendo énfasis en sus interfaces sencillas y extensibles [16].
- **Numpy**: Libreria de código de libre utilizada para el procesamiento de grandes vectores o matrices multidimensionales. Creada en el 2005, está escrita en C y Python [17].
- **Numba**: Compilador *just in time* de código libre utilizada para la optimización de código de Numpy en Python. Fue creada en el 2012 por Anaconda con el soporte de DARPA, the Gordon and Betty Moore Foundation, Intel, nvidia, AMD, y la comunidad de Github [18].
- **GTK**: Herramienta de código libre multi plataforma para la creación de interfaces gráficas [19].
- **OpenCV**: Librería de código libre con funciones para el procesado de computación de imágenes en tiempo real. Originalmente creada por Intel. Tiene soporte oficial a librerías de aprendizaje automático como Tensorflow. Inicialmente desarrollada en junio del 2000 [20].
- **MedPy** es una biblioteca y colección de scripts de codigo libre para el procesamiento de imágenes médicas en Python. Contiene funcionalidades básicas para leer, escribir y manipular imágenes grandes de dimensiones arbitrarias [21].
- **CUDA**: Es una plataforma de cálculo paralelo y un modelo de programación desarrollado por NVIDIA para la computación general en unidades de procesamiento gráfico (GPU) [22].

3.1.3 Entornos de desarrollo

Durante el desarrollo del proyecto se han utilizado los siguientes entornos:

- **Google colabority**: Es un entorno *Jupyter notebook* que no requiere de configuración y que corre enteramente en la nube creado por Google [23].
  Jupyter notebook: Son documentos JSON con un seguimiento de versiones y contienen una lista de entradas y salidas ordenadas por celdas que contienen fragmentos de código o texto. Suelen terminar con la extensión *.ipynb* [24].
- **Local**: Ordenador local utilizado durante el proyecto. Sus características son las siguientes:
  - Procesador: Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz 3.30GHz

- Memoria RAM: 8.0 GB
- Sistema operativo: Windows 10 (64 bits)
- Tarjeta gráfica: NVIDIA GeForce GTX 960, 6034 MB

3.1.4 Base de datos

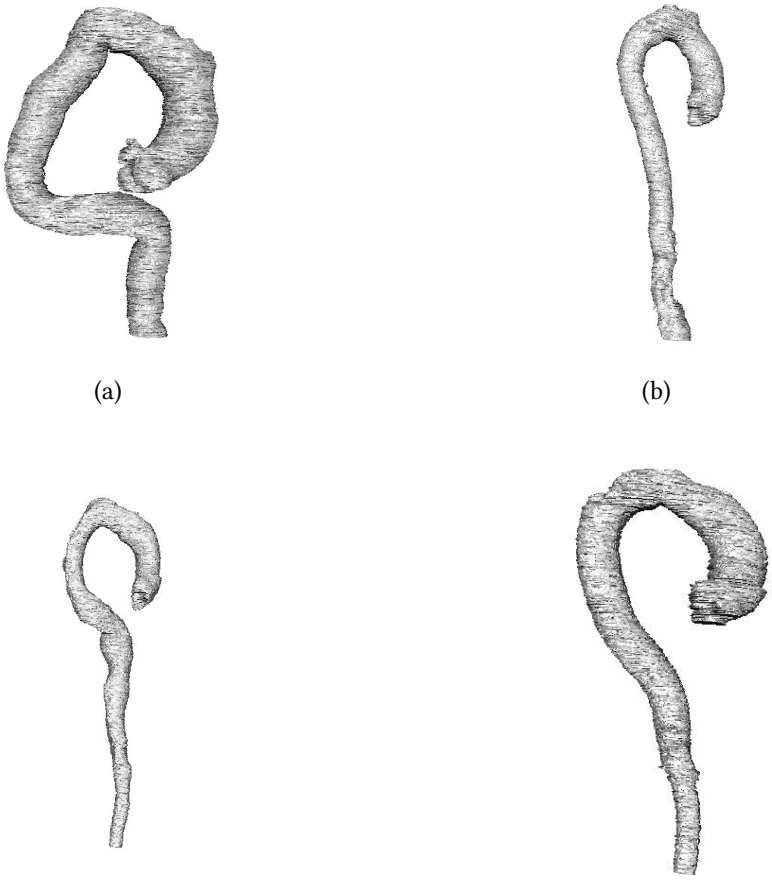
La base de datos empleada ha sido una parte de la base de datos utilizada en un artículo sobre segmentación de la aorta, publicada por la Universidad de Santiago [3]. Esta base de datos consta de 10 casos (3986 imágenes), almacenados en formato Analyze y sus respectivas máscaras de aorta, marcadas por un radiólogo (Figura 3.1).
Analyze es uno de los formatos más populares en imagen médica. Se compone de dos ficheros [25]:

- Fichero imagen: Píxeles no comprimidos de la imagen en uno de sus posibles formatos posibles (JPG,TIFF...).
- Fichero cabecera: representado como una estructura C donde guarda toda la información relevante sobre las dimensiones y los formatos de la imagen.

3.2 Tecnologías de documentación

La lista de tecnologías utilizadas para la documentación y control de versiones a lo largo del proyecto son las siguientes:

- **Latex**: Software libre de composición tipográfica de alta calidad. Incluye características diseñadas para la producción de documentación técnica y científica. Es el estándar de facto para la comunicación y publicación de documentos científicos [26].
- **Google Drive**: Sistema de almacenamiento online creado por Google [27].
- **Git**: Es un sistema distribuido de control de versiones para el seguimiento de los cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 para el desarrollo del núcleo de Linux, con otros desarrolladores que contribuyeron a su desarrollo inicial [28].
- **Draw.io**: Herramienta de código libre para la construcción de diagramas [29].



(a) (b)

(c) (d)

Figura 3.1: **Segmentaciones de la aorta realizadas por un radiólogo**: a) Caso 119. b) Caso 139. c) Caso 173. d) Caso 164.

Capítulo 4

Metodología de desarrollo

En este capítulo se explicará la planificación del proyecto y cada una de sus fases o iteraciones así como la gestión del mismo describiendo su estimación y su gestión de riesgos.

4.1 SCRUM

Scrum es un marco para la gestión del trabajo de conocimiento con principal énfasis en el desarrollo de software. Divide el proyecto en una serie de *sprints* o iteraciones que se ocupan de que las distintas partes del código estén codificadas y testeadas permitiendo la agilidad del proyecto. Se recomienda su uso para grupos de entre tres y nueve personas con reuniones de menos de quince minutos entre cada una de las iteraciones [30]. A continuación se describen las iteraciones en las que se ha dividido el proyecto.

4.1.1 Primera iteración: Limpieza de datos

La primera iteración consiste en reunir, limpiar y generalizar la base de datos para poder entrenar la red. Además, en esta iteración se adaptará el entorno de trabajo para el desarrollo del proyecto. Consta de las siguientes fases:

- **Análisis**: En esta fase se analizará la base de datos y se creará el software especializado para poder manipularla según la necesidades requeridas.
- **Diseño**: Se diseñará una librería de funciones que permita trabajar con los casos incluidos de base de datos y representarlos gráficamente. Se utilizarán los formatos originales (HDR/IMG) [25] habituales en imagen médica y se convertirán a formatos estándares en ingeniería de datos. Se adaptará el entorno de trabajo a las herramientas usadas en este proyecto.
- **Implementación** de las funciones principales de la librería.

- **Pruebas**: Se comprobará que el entorno funciona adecuadamente y la nueva base de datos está bien definida.

4.1.2 Segunda iteración: Creación modelo y preprocesado.

Está iteración se ocupa del entrenamiento de los modelos y diseño del preprocesado de datos de entrada. Consta de las siguientes fases:

- **Análisis**: En esta fase se analizará las características de la red y sus dados de entrada.
- **Diseño** de los distintos escenarios de entrenamiento, en los que se hace énfasis en los dos siguientes puntos:
  - Las modificaciones de la propia red UNet para mejorar su precisión y rendimiento en este proyecto.
  - Las modificaciones en el preprocesado de los datos entrantes para mayor optimización del uso de la red.
- **Implementación**: Esta fase consiste en la codificación y ejecución de los modelos y las fases de preprocesado.
- **Pruebas** de precisión de la la propia red.

4.1.3 Tercera iteración: Post Procesado

Esta iteración se ocupado de limpiar el ruido sobrante tras la ejecución de la red. Consta de las siguientes fases:

- **Análisis**: En esta fase se analizará los resultados obtenidos para buscar estrategias óptimas que resuelvan los problemas que no pudieron solucionarse en la iteración anterior.
- **Diseño** de los distintos métodos posibles para limpiar el ruido obtenido.
- **Implementación** de los métodos diseñados en la fase de diseño.
- **Pruebas**: Testeo de los mismos para decidir cual es el mejor para realizar la tarea.

4.1.4 Cuarta iteración: Creación de la interfaz gráfica

En esta iteración se aborda la creación de un programa que a partir de los datos de entrada genere la segmentación y visualice el resultado. Consta de las siguientes fases:

- **Análisis**: En esta fase se analizará que necesidades que debe cumplir la interfaz para facilitar el uso de su funcionalidad de cara al usuario.

- **Diseño** de la aplicación y de su interfaz.
- **Implementación** de la aplicación.
- **Pruebas** de sistema y usabilidad para asegurarse su correcto funcionamiento.

4.2 Planificación

La estimación total del proyecto fue de nueve meses desde su inicio en septiembre hasta su finalización en mayo. Todas las iteraciones a excepción de la segunda serán de una duración de dos meses, esta iteración es más larga dado que es la parte principal del sistema y la que requiere más tiempo. A la hora de organizar el proyecto se tomaron en cuenta las siguientes características:

- Cada etapa requiere de la anterior por lo que para empezar una nueva iteración primero se debía finalizar la anterior.
- Las tareas de cada una de las tres primeras iteraciones son paralelas entre si dado la naturaleza del proyecto, se necesita trabajar con ellas al mismo tiempo.
- En la cuarta iteración la codificación y el testeo se hacen al mismo tiempo por falta de tiempo.

Esta estimación esta desarrollada para el trabajo de una persona. En la Figura 4.1 se puede observar como esta el proyecto organizado en el diagrama de Gantt.

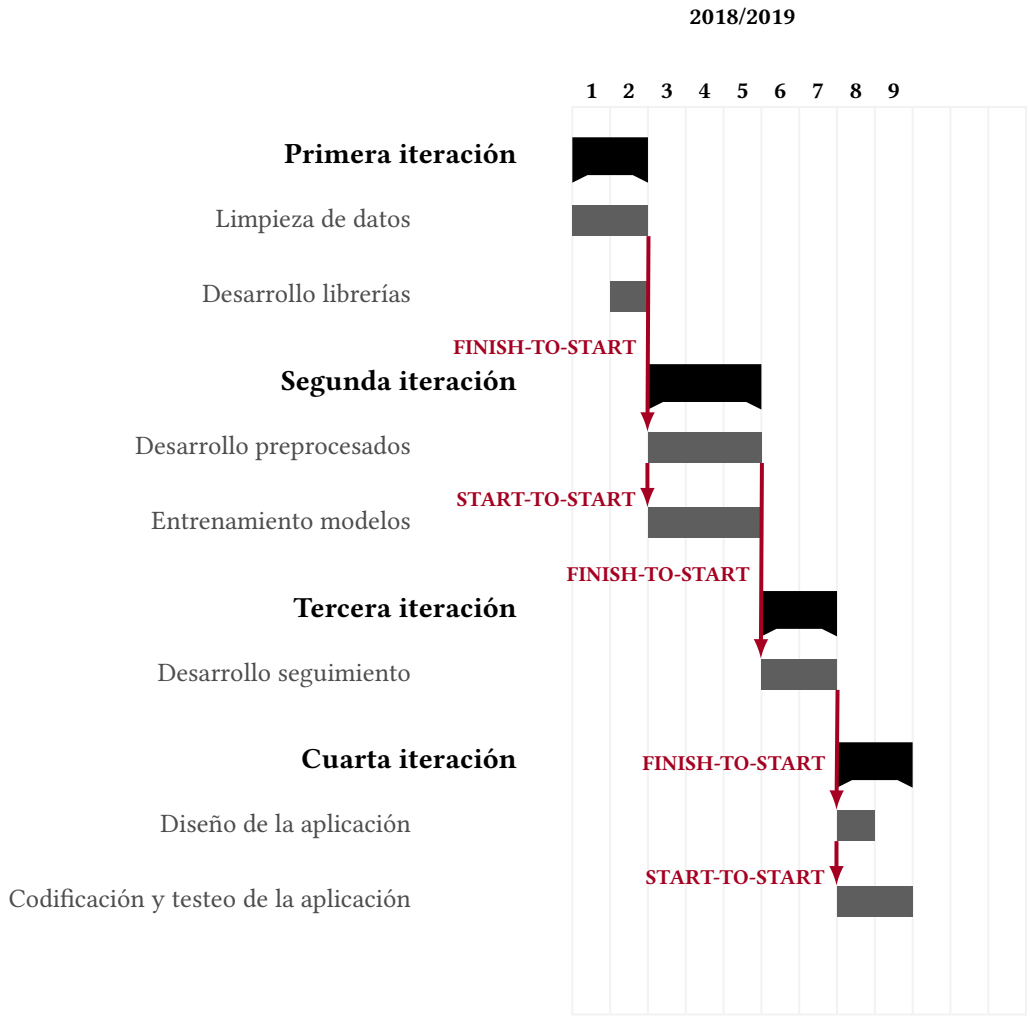


Figura 4.1: **Diagrama de Gantt** de la planificación del proyecto

4.3 Seguimiento

El proyecto acabó alargándose a diez meses dado a la complejidad del problema. Los modelos desarrollados en la segunda iteración por problemas técnicos de su arquitectura hicieron que el postprocesado tuviera que ser más complejo del esperado en la estimación inicial. Esto llevó al alargamiento de la tercera iteración a tres meses dado que el método de postprocesado de la aorta se tuvo que partir en dos métodos distintos para poder hacer una óptima segmentación.

Esto hizo que se retrasara el comienzo de la cuarta iteración un mes más de lo previsto. En la Figura 4.2 se puede observar el diagrama de Gantt con el seguimiento realizado.

Tabla 4.1: **Tabla de riesgos del proyecto.**

Riesgo	Valor	Probabilidad	Total
Escasez de capacidad computacional	4	4	4
Borrado accidental del software desarrollado	5	2	3
Perdida de la base de datos	3	1	2
Incompatibilidad entre tecnologías	4	1	2
Falta de tiempo para el proyecto	5	4	4

4.4 Gestión de riesgos

En este proyecto se tomaron en cuenta una serie de riesgos que pudieran afectar el desarrollo del proyecto. Para poder calcular su relevancia se les asignaron tres valores:

- **Valor**: Asigna la importancia y el efecto que tendría si el riesgo ocurre.
- **Probabilidad**: La probabilidad de que ese riesgo ocurra.
- **Total**: La importancia total del riesgo. Se calcula a partir de los dos parámetros anteriores:

$$\textbf{Total} = \textbf{Valor} * 0.5 + \textbf{Probabilidad} * 0.5 \tag{4.1}$$

Los tres valores tienen el mismo rango de valores siendo 1 el más pequeño y 5 el más grande, todos ellos son números enteros.
Como se puede ver en la tabla de riesgos (Tabla 4.1) los principales riesgos del proyecto son la escasez de la capacidad computacional, esto es debido a la gran cantidad de operaciones necesarias para el entrenamiento de las redes de neuronas y la falta de tiempo para el proyecto. Siendo la segunda debido a que al ser un proyecto de investigación es difícil estimar el tiempo necesario para obtener los resultados previstos inicialmente.

4.5 Estimación de costes

A continuación se muestra la estimación realizada sobre los costes de desarrollo del proyecto. Se ha determinado la existencia de dos perfiles: el jefe de proyecto y el desarrollador. Se estima que el trabajo de un desarrollador novel puede costar de media 18€/hora. En cuanto al trabajo del jefe de proyecto se ha estimado en 45 €/hora.

Estimando que de media se ha trabajado 5 horas al día, en la Tabla 4.2 se muestran los costes estimados del trabajo del desarrollador. Suponiendo que el jefe de proyecto ha dedicado 200 horas a éste, obtenemos un precio estimado total del proyecto de 25.200,00 € (Tabla 4.3).

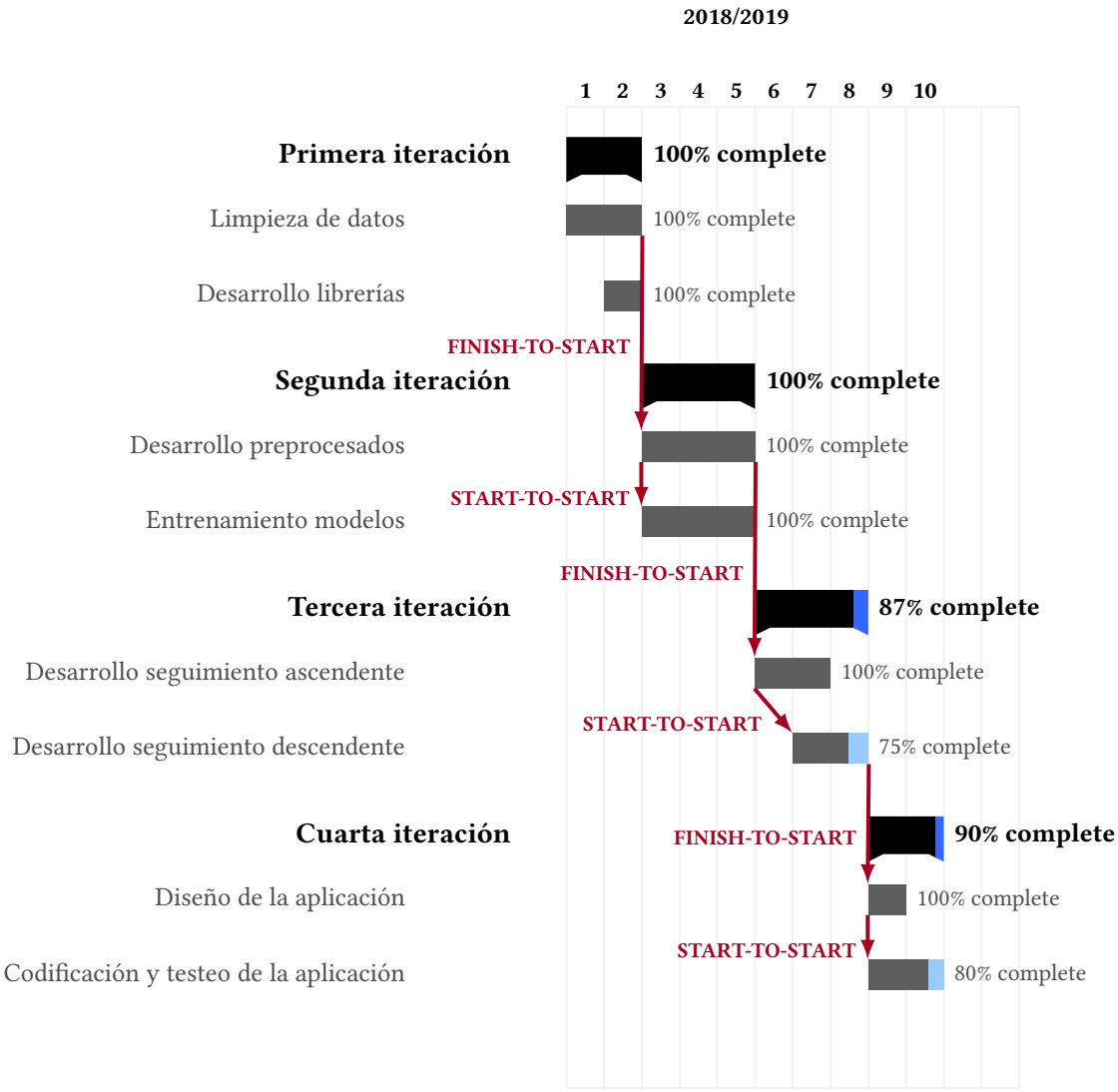


Figura 4.2: **Diagrama de Gantt** del seguimiento del proyecto

Tabla 4.2: Estimación coste del desarrollador.

Iteración	Horas de trabajo	Coste
Limpieza de datos	200	3.600,00€
Creación modelo y preprocesado	300	5.400,00€
Post Procesado	200	3.600,00€
Creación de la interfaz gráfica	200	3.600,00€

Tabla 4.3: Estimación coste total.

Perfil	**Coste**
Desarrollador	16.200,00€
Jefe proyecto	9.000,00€
Total	25.200,00€

Capítulo 5

Método propuesto

En este capítulo se explicará los distintos métodos propuesto para la resolución del proyecto. El proyecto se divide en preprocesado de la imagen, segmentación de la estructura de la aorta y postprocesado de los resultados de la segmentación.

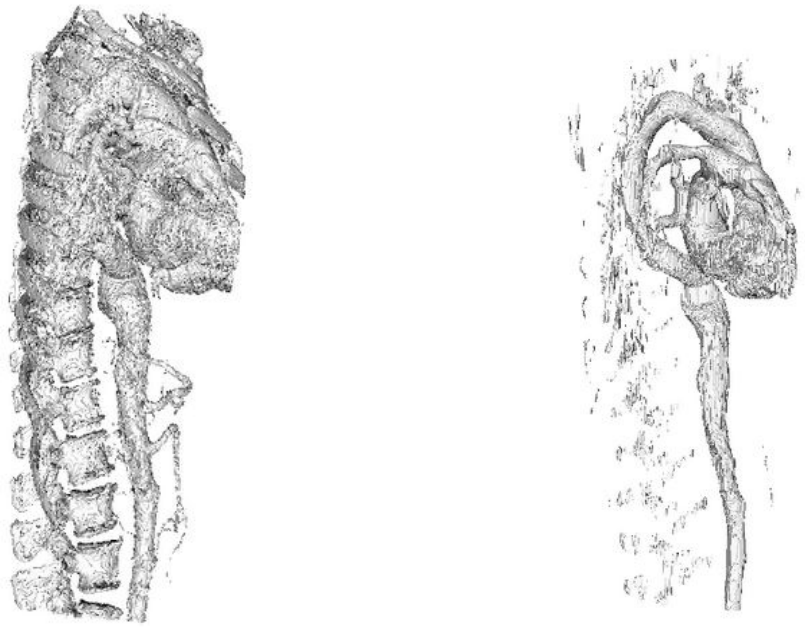
5.1 Métodos de preprocesado

Estos métodos se encargarán de simplificar la información recibida por la red, eliminando parte del ruido y remarcando la información relevante.

El objetivo de las funciones de preprocesado es facilitar el trabajo de procesado del modelo usado. En el proyecto actual se intentó principalmente filtrar las imágenes por su rango de colores. Esto es debido a que las imágenes CT usan la escala Hounsfield (Apéndice B) que permite aislar estructuras del mismo tipo según su intensidad.

Cada una de las distintas normalizaciones propuestas fue diseñada para realzar la forma o los valores de nivel de gris de la aorta. Podemos ver un ejemplo de normalización en la Figura 5.2 donde se observa como tomando ciertos valores es posible eliminar gran parte de de las estructuras. Sin embargo, las operaciones de preprocesado no pueden ser estrictas dado que por cada caso el rango de valores que forma la aorta varía, así que lo que puede ser bueno para un caso puede no valer para otro. Por eso las reglas de preprocesado deben hacer un filtrado de información generalizado dejando el trabajo más especializado a la red. Los métodos de preprocesado sobre la imagen original propuestos para la segmentación son los siguientes:

- **Imagen original**: La entrada de la red son imágenes sin modificaciones. Su rango de valores es [-1024,3072] en escala Hounsfield (Figura 5.2 a).
- **Imagen binaria**: La entrada a la red es una imagen binaria (0, 1) donde los 1s son los píxeles que estén dentro del valor del rango establecido. Su función principal es que ignore los colores y solo se fije en la forma de la aorta. La ecuación utilizada sería la



(a) (b)

Figura 5.1: Ejemplo de preprocesado. a) Parte de las estructuras de un caso sin preprocesar. b) Estructuras preprocesadas tras aplicar la etapa de preprocesado.

siguiente:

$$f(I(i,j)) = \begin{cases} 1 & Si \quad \beta \leq I(i,j) \leq \alpha, \\ 0 & resto \end{cases} \tag{5.1}$$

donde i e j son las coordenadas de la imagen I, α es el límite máximo y β, el mínimo (Figura 5.2 b).

- **Imagen lineal**: La entrada es una imagen filtrada, al igual que la binaria, pero los valores de los píxeles que están en el rango fueron fueron modificados a un nuevo rango [0.0,1.0]. El objetivo era normalizar la imagen teniendo sus valores concentrados:

$$f(I(i,j)) = \begin{cases} m \cdot (I(i,j) - z) & Si \quad \beta \leq I(i,j) \leq \alpha, \\ 0 & resto \end{cases} \tag{5.2}$$

donde i y j son las coordenadas de la imagen I, α es el límite máximo, β, el mínimo del rango propuesto y m, z son los parámetros para generar una recta (Figura 5.2 c).

- **Imagen original filtrada**: La entrada es una imagen filtrada al igual que la binaria pero los valores de los píxeles que están dentro del rango mantienen su valor original. Su objetivo es similar al preprocesado original pero dejando los valores en sus posiciones

originales:

$$f(I(i,j)) = \begin{cases} I(i,j) & Si \quad \beta \leq I(i,j) \leq \alpha, \\ 0 & resto \end{cases} \tag{5.3}$$

donde i y j son las coordenadas de la imagen I, α es el límite máximo y β, el mínimo del rango propuesto (Figura 5.2 d).

- **Imagen binaria concatenada**: Se filtran los valores de los píxeles en un rango, los que están fuera del rango mantienen su valor mientras que los que están dentro su valor aumenta a n, siendo n un valor alto. Su función es la de resaltar los valores relevantes para la red pero dejar el resto visibles:

$$f(I(i,j)) = \begin{cases} n & Si \quad \beta \leq I(i,j) \leq \alpha, \\ I(i,j) & resto \end{cases} \tag{5.4}$$

donde i y j son las coordenadas de la imagen I, α es el límite máximo y β, el mínimo del rango propuesto (Figura 5.2 e).

- **Imagen aumentada**: Muy similar a caso anterior pero en vez de aumentar los valores a n, los valores se concatenan con la suma del valor del píxel más una constante. Es la misma idea que en el caso anterior pero intentando mantener información sobre los valores originales de los valores aumentados de los píxeles:

$$f(I(i,j)) = \begin{cases} I(i,j) + n & Si \quad \beta \leq I(i,j) \leq \alpha, \\ I(i,j) & resto \end{cases} \tag{5.5}$$

donde i y j son las coordenadas de la imagen, α es el límite máximo y β, el mínimo (Figura 5.2 f).

- **Imagen con profundidad**: Suma de varias imágenes binarias consecutivas con el fin de obtener la profundidad de los distintos elementos de la imagen. De esta forma la red puede conocer la profundidad de cada uno de los elementos y poder analizar varias imágenes en una con la siguiente ecuación:

$$f(I(i,j)) = \sum_{k=0}^{n} f_b(I_k(i,j)) \tag{5.6}$$

$$f_b(I(i,j)) = \begin{cases} 1 & Si \quad \beta \leq I(i,j) \leq \alpha, \\ 0 & resto \end{cases} \tag{5.7}$$

donde i y j son las coordenadas de la imagen I, α es el límite máximo y β, el mínimo del rango propuesto (Figura 5.2 g).

5.2 Modelos

Todos los modelos de *deep learning* testeados comparten la arquitectura *UNet* con ligeras variantes con el fin de encontrar la mejor para la segmentación de la aorta. Los modelos están agrupados en generaciones, cada una de ellas agrupa las redes neuronales entrenadas en un periodo de tiempo determinado. Los cambios realizados en las primeras generaciones fueron más generales, mientras que en las últimas eran mucho más específicos. El ciclo de cada generación es el siguiente:

1. **Analizar** los resultados obtenidos en generaciones anteriores para poder deducir que mejoras implementar y que línea seguir a la hora de diseñar nuevos modelos.

2. **Crear y entrenar** modelos nuevos con el fin de mejorar los resultados obtenidos anteriormente. En general, los tipos de modificaciones realizadas a los modelos de una generación comparten similitudes tales como modificar el número de capas o cambiar funciones de activación.

3. **Documentar** los resultados y observaciones de cada uno de los modelos.

5.2.1 Entrenamiento

Una de las partes más importantes a la hora de obtener buenos resultados en un modelo es la de entrenamiento. Parámetros como el número de datos por sample *batch size* o el número de iteraciones de entrenamiento *epochs* pueden ser claves a la hora de que la red funcione correctamente. Los parámetros de cada entrenamiento se cambian en cada generación, por lo que nunca se usan valores constantes. Los entrenamientos presentaron dificultades debido a la cantidad de memoria *RAM* necesaria para el proceso, para solucionar este problema se hizo uso de un generador (Apéndice C). No se uso técnicas de *data augmentation* debido a que distorsionaban la imagen y la posición de las distintas estructuras en ella perjudicando el modelo.

5.2.2 Descripción modelos

Los modelos de cada generación fueron enumerados de la siguiente manera:
model_(número generación)_(número subsección generación, opcional)_(letra alfabeto)

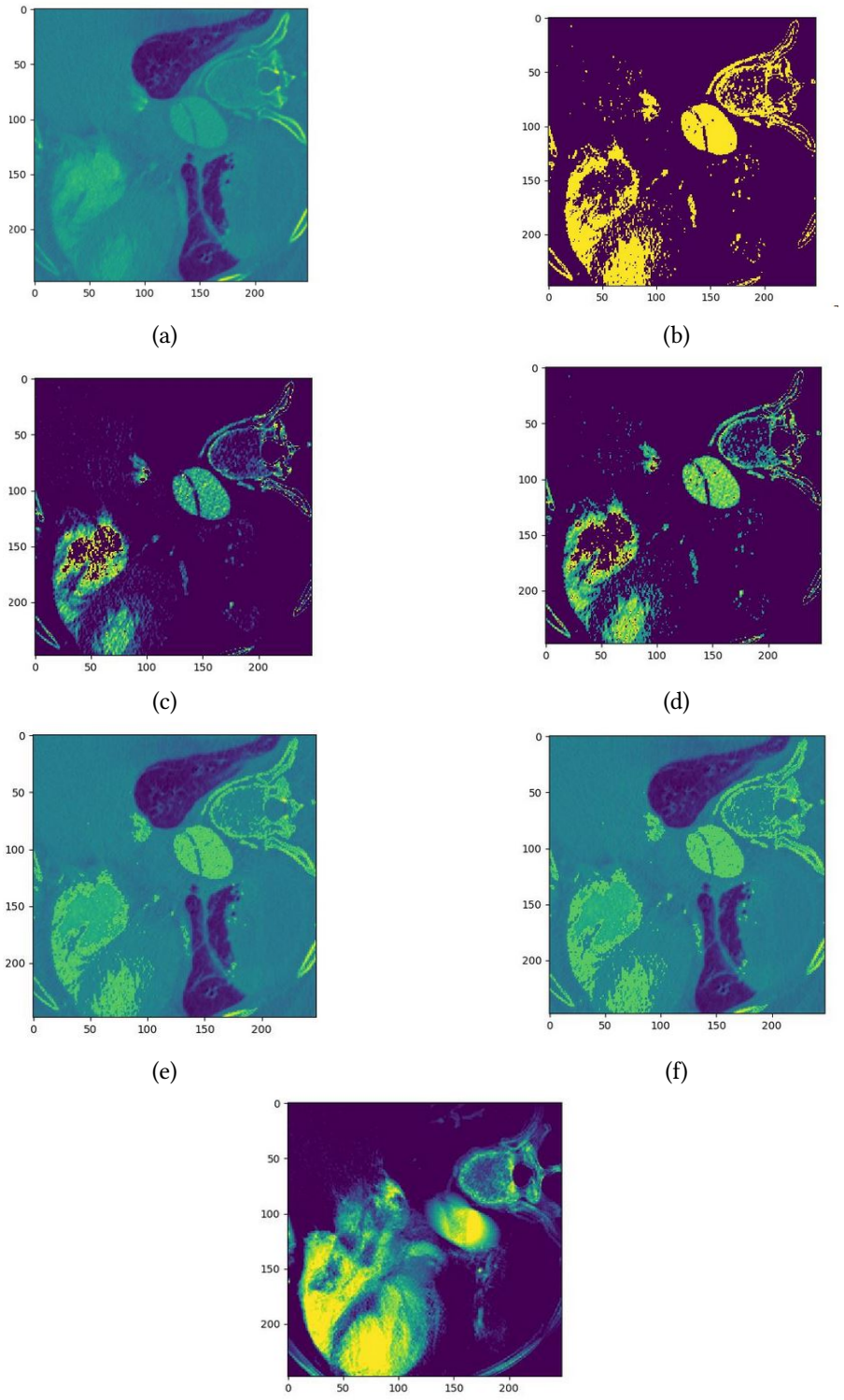


Figura 5.2: **Métodos de preprocesado**: a) Imagen original .b) Imagen binaria .c) Imagen lineal .d) Imagen original filtrada. e) Imagen binaria concatenada. f) Imagen aumentada. g) Imagen profunda.

de esta forma se puede realizar un seguimiento de los mismos (por ejemplo, modelo_7_a, modelo_6_2_c), se desarrollaron un total de ocho generaciones de modelos *UNet*:

- **Primeras cuatro generaciones**: Fueron las generaciones más cortas en número de modelos por generación. Se trató de identificar los puntos más relevantes en el proceso como cuales eran los mejores métodos de preprocesado, funciones de activación y funciones de optimización así como localizar dificultades que se podían presentar a la red.

- **Quinta generación**: En esta generación se analizaron los resultados de las anteriores y se establecieron las siguientes características para los futuros modelos:

  - Todas las funciones de activación debían ser *ReLU*.
  - La mejor función de optimización para la segmentación sería *Adam*.
  - El método de preprocesado que obtiene mejores resultados es el de *Imagen original filtrada*.

  En esta generación fue en la que mayor número de modelos fueron entrenados dado que se quería testear cuál era el mejor diseño de la red para la segmentación. Para ello se fueron probando distintos número de capas así como diferentes tamaños y cantidades de filtros.

- **Sexta generación**: Se empezó a usar *DSC* para la medición de precisión ya que era el método más adecuado para valorar los resultados obtenidos en este tipo de problemas. La generación se probaron distintas funciones de *loss*:

  - **Modelos 6.1** la función de *loss* es el negativo de DSC:

  $$funcion\_loss = -DSC \tag{5.8}$$

  - **Modelos 6.2**: la función de *loss* es el opuesto de *DSC*:

  $$funcion\_loss = 1 - DSC \tag{5.9}$$

  - **Modelos 6.3** se testeó *Jaccard* como función *loss*.

- **Séptima generación**: El motivo de esta generación fue el de testear el método de preprocesado *Imagen con profundidad* como una posible mejora.

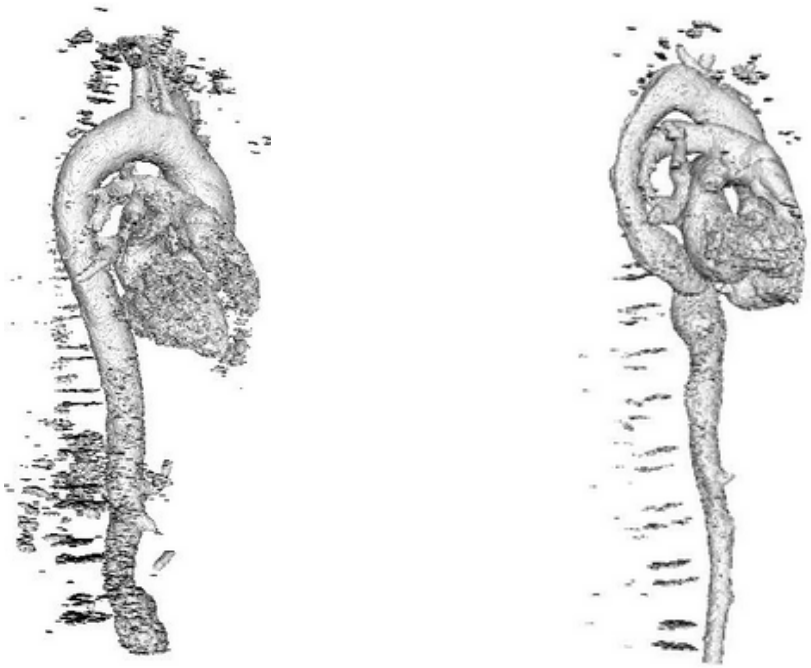
- **Octava generación**: Las imágenes que recibe la red contienen una gran cantidad de ruido que se puede eliminar o reducir con operaciones morfológicas (Apéndice D). Sin

embargo, en vez de realizar esa tarea en el preprocesado se decidió entrenar la red para optimizar el proceso. Así, el proceso de entrenamiento sería el siguiente:

1. Entrenar la red de manera habitual usando arquitecturas de la sexta generación.
2. Usar funciones morfológicas de reducción en los conjuntos de entrenamiento, validación y testeo. Volver a entrenar la red.

Es importante usar los mismos conjuntos en ambas iteraciones de entrenamiento, si se mezclan podemos obtener resultados erróneos o obtener modelos sobreentrenados.

Entre las generaciones seis y siete se desarrolló un método alternativo de segmentación basado en operaciones morfológicas. Su nomenclatura es la siguiente:
model _m _(letra alfabeto)
Los resultados obtenidos en esta generación fueron los que precedieron al diseño de los modelos de la octava generación (Figura 5.3).



(a) (b)

Figura 5.3: **Segmentaciones obtenidas de los modelos de segmentación basado en operaciones morfológicas**: a) Caso 139. b) Caso 164.

5.3 Seguimiento de la aorta en 3D

Como se ha explicado la presencia del corazón en el modelo presenta múltiples dificultades en razón de su semejanza, en color y forma, con la propia aorta. Para solucionar este problema, se presentan distintas ideas con las que pretendía hacer un seguimiento de la estructura de la aorta hasta el arco aórtico. La aplicación de este tipo de técnicas permitió la eliminación de otros posibles factores de ruido que hayan sido segmentados por la UNet como algún trozo óseo.

La entrada de estos algoritmos de seguimiento es la salida de la UNet:

$$O_k(i,j) = \begin{cases} 1 & Si \quad I_k(i,j) \in aorta, \\ 0 & resto \end{cases} \tag{5.10}$$

donde k es la capa de la imagen, i y j las coordenadas de la capa I_k. O es un array de tres dimensiones binario, en el que se representan como 0 los píxeles de fondo y como 1 las posibles partes de la aorta.

A continuación explicamos varias alternativas para realizar el seguimiento de la aorta hasta el arco aórtico.

5.3.1 Proyección y seguimiento de puntos

Este algoritmo de seguimiento presenta una forma sencilla de implementación. Está dividido en tres pasos:

1. **Proyección capa inferior**: Se proyecta la capa inferior a la capa actual siguiendo la siguiente expresión:

$$P_k(i,j) = \begin{cases} n & Si \quad I_k(i,j) \cap P_{k-1}(i,j) \neq 0, \\ P_{k-1}(i,j) & resto \end{cases} \tag{5.11}$$

   donde n es es valor que queremos asignar en la proyección e (i, j) son las coordenadas de la imagen I_k y P_{k-1} representa los puntos proyectados.

2. **Desplazamiento capa actual**: Siendo O_k conjunto de puntos no cero de la capa k y P_k los proyectados se seleccionan los puntos que devuelve la siguiente ecuación:

$$D_k = O_k - P_k \tag{5.12}$$

3. **Seguimiento de puntos**: Busca los puntos del conjunto D_k que están unidos a algún

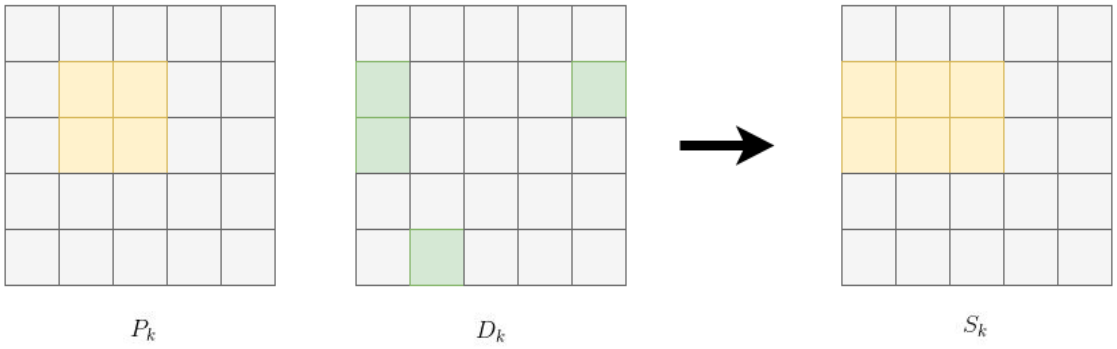


Figura 5.4: **Seguimiento de puntos**: En este paso se localizan los puntos del conjunto D_k que están unidos a algún punto del conjunto P_k.

punto del conjunto P_k (Figura 5.4).

$$S_k(i,j) = \begin{cases} 1 & Si \quad P_k(i,j) = 1 \cup D_k(l,m) = 1 \cap d(l,m) \in N(i,j), \\ 0 & resto \end{cases} \tag{5.13}$$

donde $N(i,j)$ es el conjunto de 8 píxeles vecinos del píxel (i, j).

La Figura 5.5 muestra los principales pasos de este algoritmo sobre una imagen de ejemplo.

El algoritmo *proyección y seguimiento de puntos* consigue dar resultados buenos (Figura 5.6) pero presenta una serie de problemas. En primer lugar es demasiado lento, llegando a tardar más de 50 segundos por caso. En segundo lugar, no fuerza el mantenimiento de una forma determinada, por lo que si el corazón u otro tipo de ruido que se intenta eliminar está pegado a la aorta, el algoritmo lo detectará como parte de la misma. Otro problema que presenta es que es necesario tomar la capa inicial como segmentación correcta, dado que no se puede comprobar si hay ruido en ella o no. Además, sumado a esto, es difícil de paralelizar lo que complica su optimización.

5.3.2 Seguimiento paralelizable

Este algoritmo de seguimiento se creó con el fin de ser fácilmente paralelizable y así poder reducir los tiempos de ejecución. Su funcionamiento consta de tres partes (Figura 5.7):

1. **Buscar estructuras**: Consiste en buscar en cada capa las distintas estructuras que hay en ella. Obtiene por cada estructura las coordenadas del centro (i,j) y las coordenadas $[i_{max}, j_{max}, j_{min}, i_{min}]$ que enmarcan todos los píxeles que forman el perímetro.

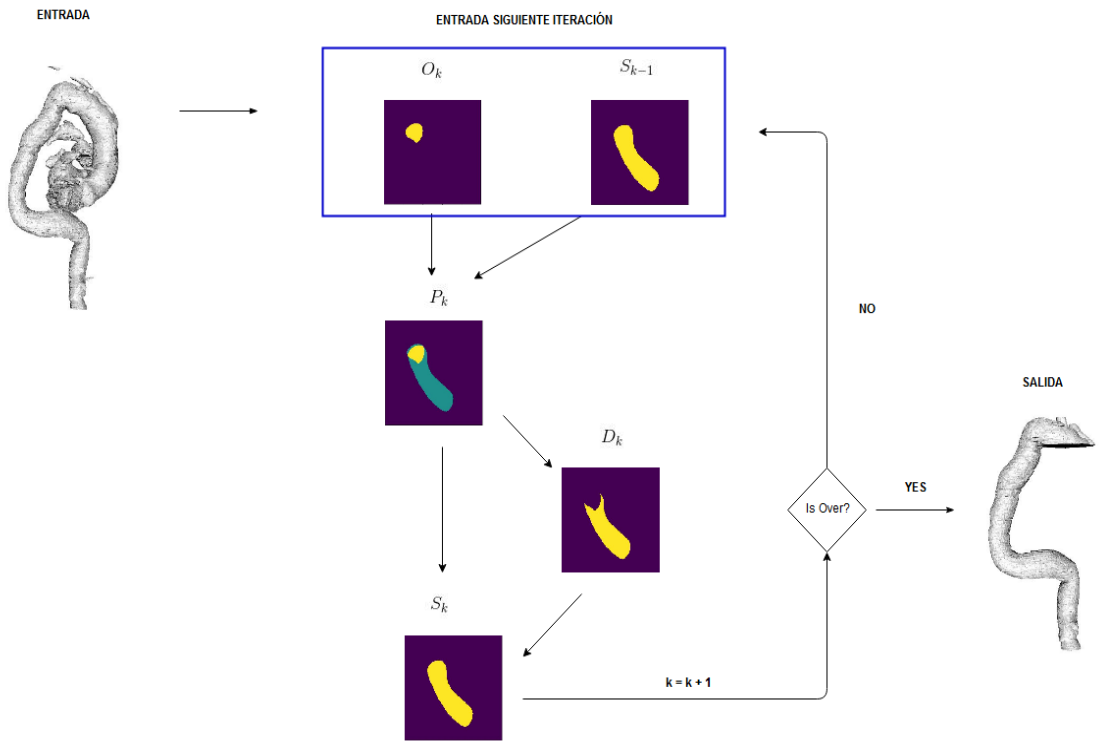


Figura 5.5: **Proyección y seguimiento de puntos**: algoritmo usado para el seguimiento de la aorta.

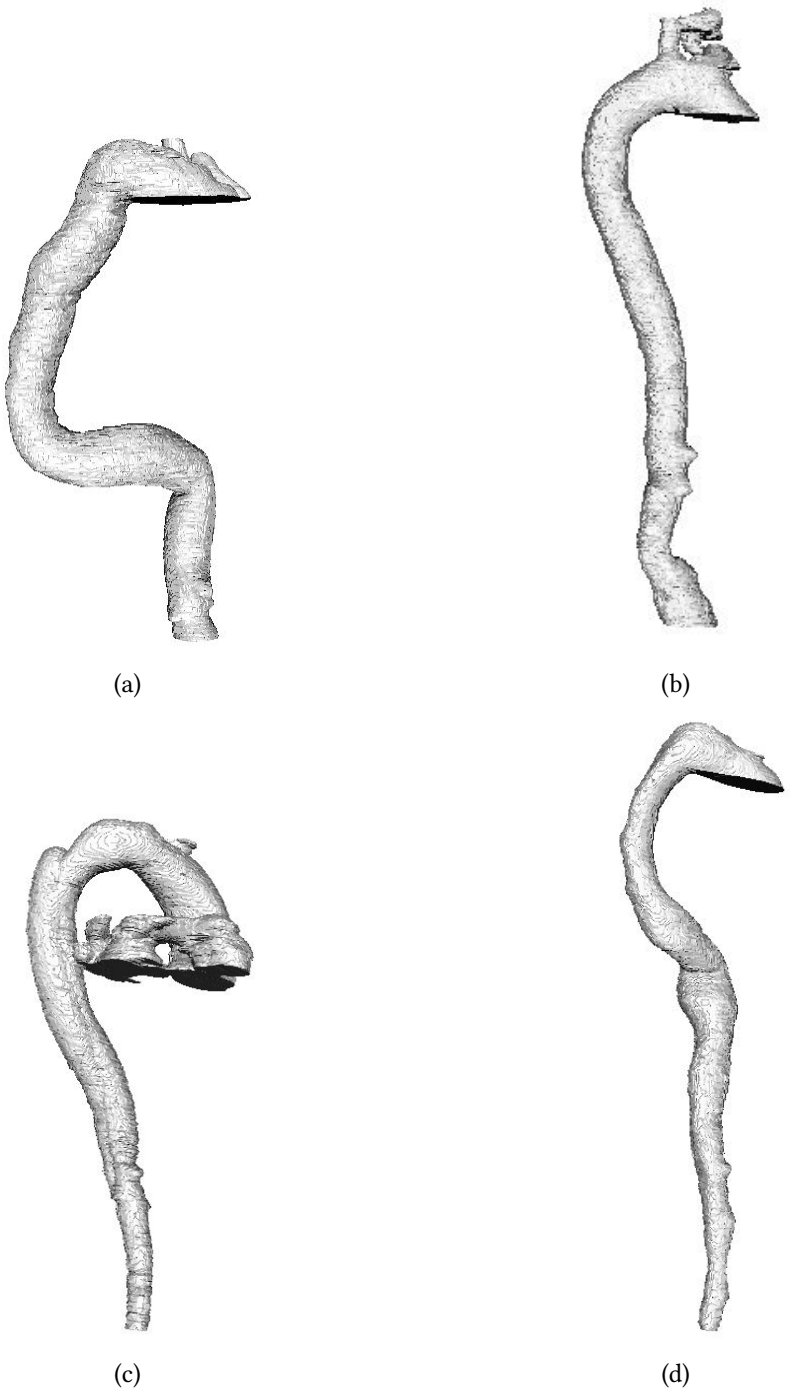


Figura 5.6: **Resultados finales de casos segmentados con Proyección y seguimiento de puntos**: a) Caso 119. b) Caso 139. c) Caso 173. d) Caso 164.

2. **Juntar estructuras**: Se fusionan las estructuras en conjuntos de tal forma que al menos todas estructuras tengan una estructura con la que satisfagan las siguientes reglas:

   - **Regla 1**: Las estructuras A,B deben compartir mismas coordenadas i y j. Esto ocurre si sus respectivos perímetros P delimitado por sus coordenadas $[i_{max}, j_{max}, j_{min}, i_{min}]$ cumplen el siguiente requisito: $A_p \cap B_p \neq \varnothing$
   - **Regla 2**: Que las estructuras A,B estén a una capa de diferencia. Siendo k el nivel de la capa se debe cumplir que $abs(A_k - B_k) == 1$
   - **Regla 3**: Una estructura no puede cumplir las tres reglas con dos estructuras situadas en la misma capa. Si A_{k+1} o A_{k-1} cumple el primer y segundo requisito con B_k y C_k se debe hacer la siguiente selección entre B y C:
     - Seleccionar el conjunto que comparte el centroide (i,j) con A.
     - Si ninguno lo comparte, seleccionar uno al azar y descartar el otro.

3. **Seleccionar estructura** con más píxeles.

Los dos primeros pasos son paralelizables. A pesar de ser más fácil de optimizar que el método de *proyección y seguimiento de puntos* en la práctica no daba resultados mejores. Esto se debe a que al no haber tantas estructuras que observar por capa no llegaba hacer tiempos mejores. Tampoco solucionaba problemas del anterior método como el de forzar a mantener una forma.

5.4 Seguimiento en 2D a partir de PCA.

Los algoritmos de seguimiento propuestos anteriormente se caracterizan por ser mucho más lentos que la propia red UNet. Dado que es la red quien hace la gran parte de la segmentación los algoritmos de seguimientos no deberían consumir tanto tiempo. Para solucionar este problema de tiempos se recurrió a técnicas de análisis de componentes principales (PCA).

5.4.1 El análisis de componentes principales (PCA)

Es un proceso matemático para la transformación de variables. Permite reducir el número de factores que influyen sobre dichas variables simplificando su representación y procesado. Esto es posible gracias a la creación de un nuevo conjunto de variables que tienen relación con las anteriores [31].

Los siguientes algoritmos de seguimiento utilizando PCA responden a la necesidad de reducir los tiempos de ejecución simplificando el número de operaciones totales necesarias

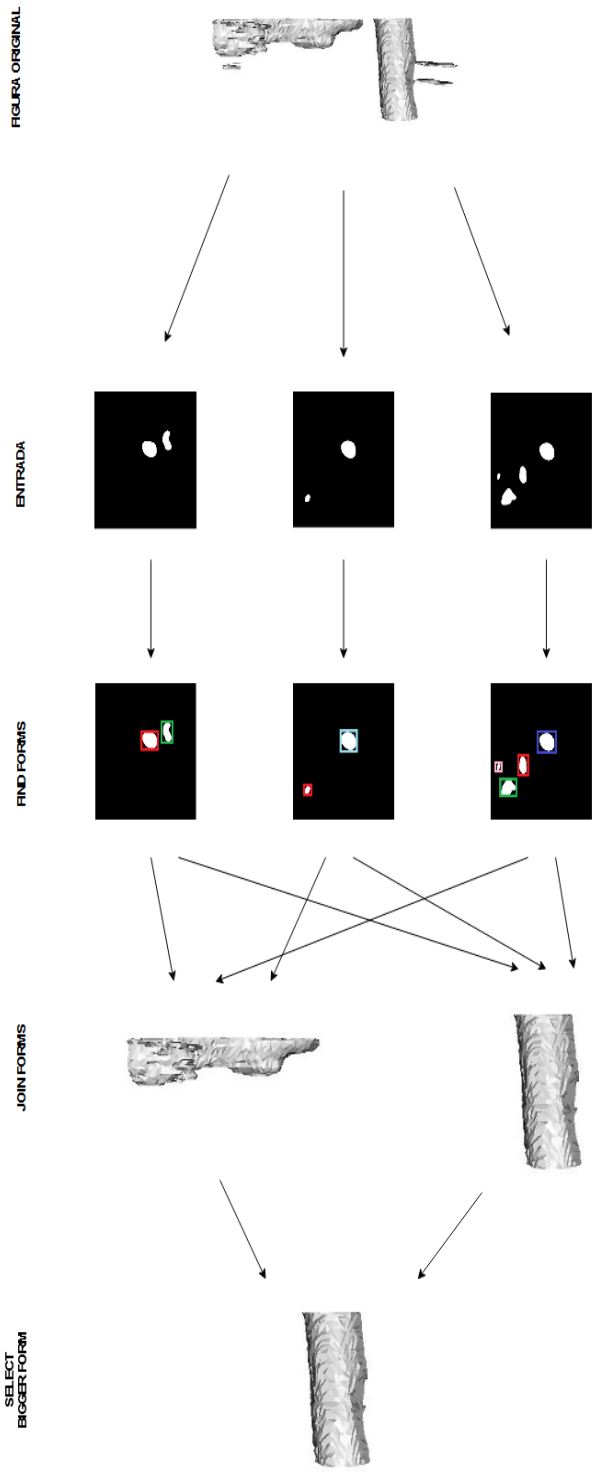


Figura 5.7: **Pasos del algoritmo de seguimiento de la aorta paralelizable**

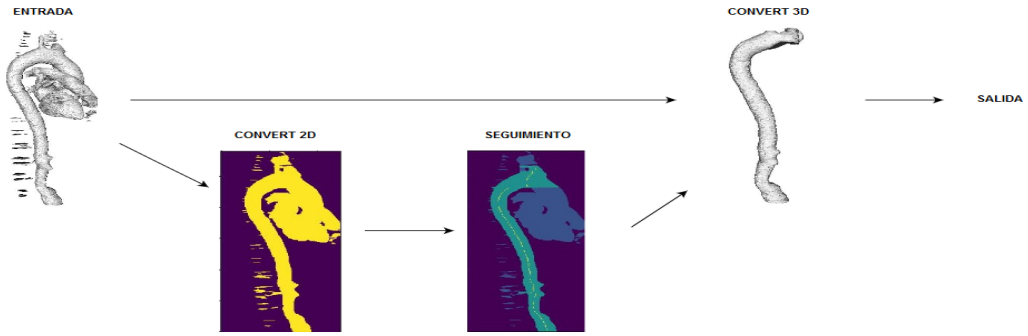


Figura 5.8: **Seguimiento 2D**: pasos a seguir en los métodos de seguimiento 2D de la aorta.

para obtener un buen resultado. Los algoritmos anteriores no sólo no conseguían hacer el sistema actual más preciso que los métodos convencionales sino que eran demasiado lentos. Por tanto, se necesita una manera distinta de abordar el problema.

La idea para optimizar los algoritmos se basa en pasar el paciente 3D a una representación 2D que tenga las mismas características y que nos permita poder volver a pasar a sus dimensiones originales de manera simple y sin perder información sobre los píxeles que formen la aorta en el proceso (Figura 5.8). Para ello se buscó una proyección en la que la aorta y las partes del corazón que no fueron eliminadas se pudieran observar de forma completa al igual que gran parte del ruido sobrante. Este método no es perfecto dado que al hacer la conversión a 2D se hace imposible eliminar el ruido detrás de la aorta teniendo que eliminarse con operaciones morfológicas cuando se vuelva a pasar a sus dimensiones originales. Sin embargo, el ruido detrás de la aorta suele ser bastante pequeño.

A continuación se explican los algoritmos propuestos para resolver el problema.

5.4.2 Profundidad 2D

Para la primera aproximación de seguimiento 2D se utilizó una conversión en la que cada píxel convertido tuviera un valor igual a la suma de píxeles que representa. La idea era ser lo más fielmente posible al paciente en 3D sin tener apenas pérdida de información en la transformación, se utilizando la siguiente conversión matemática:

$$P(i,j) = \sum_{i=0,j=0}^{n,m} I_k(i,j) \tag{5.14}$$

donde (i, j) son las coordenadas de la imagen I y n, m sus respectivos límites (Figura 5.9). Una vez obtenida la conversión se establece un rango entre la media de los valores y su máximo y se filtra el resto.

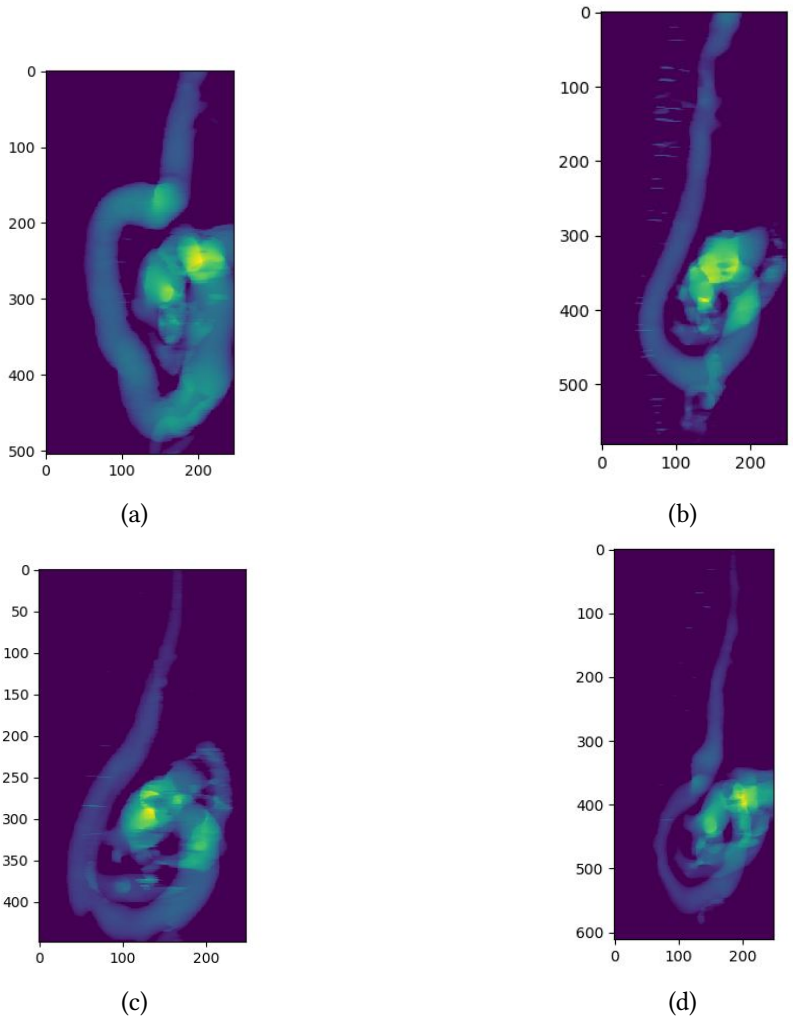


Figura 5.9: **Método profundidad 2D** conversiones a 2D: a) Caso 119. b) Caso 139. c) Caso 173. d) Caso 164.

5.4.3 Seguimiento 2D binario

Es esta segunda aproximación se realiza una conversión que devuelve valores que simbolicen la forma de la aorta sin tener en cuenta su profundidad. Los nuevos valores de los píxeles en 2D eran de carácter binario y se obtenían con la siguiente ecuación:

$$P_k(i,j) = \begin{cases} 1 & Si \quad \sum_{i=0}^{n}\sum_{j=0}^{m} I_k(i,j) \geq 1, \\ 0 & resto \end{cases} \tag{5.15}$$

donde (i,j) son las coordenadas de la imagen I_k. Utilizando esta ecuación de punto de partida se propone dos métodos para realizar el seguimiento los cuales se explican a continuación.

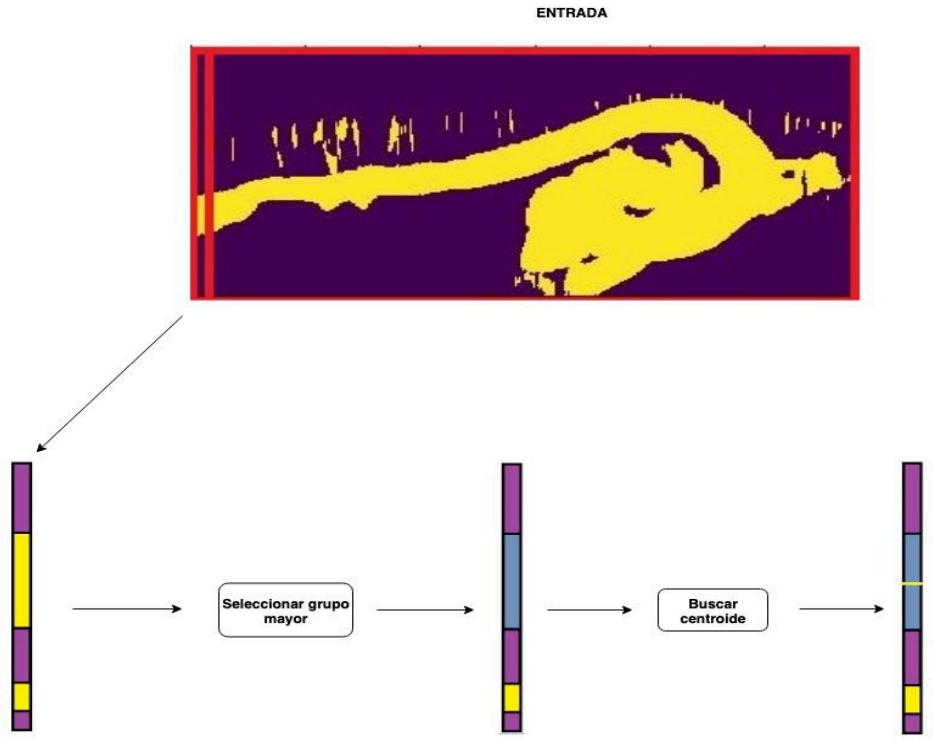


Figura 5.10: **Seguimiento de forma 2D**: procesado de la capa inicial.

Proyección y seguimiento de puntos 2D

Se desarrolló una versión 2D del algoritmo de *proyección y seguimiento de puntos*. La estructura seguía siendo la misma empezando a recorrer de abajo arriba toda la imagen 3D, solo que en esta ocasión cada capa es un array 1D correspondiente a una fila de la imagen 2D.

Para forzar que siga una estructura en concreto, en la etapa de seguimiento de puntos solo se puede seleccionar un número n de píxeles más que los que formaban la aorta de la capa anterior. Este paso se hizo fundamental dado que, al pasar los pacientes a 2D, el ruido que antes estaba separado de la aorta ahora estaba pegado a ella.

Seguimiento de forma 2D

Este método está dividido en varias fases, se basa en seguir la forma de la aorta utilizando su centroide. El algoritmo está dividido en dos partes: operar la primera capa y operar el resto de capas.

Operar la primera capa se basa en encontrar cuál de las distintas estructuras que aparecen en la capa es la de la aorta (Figura 5.10). Esta parte tiene dos etapas:

1. **Obtener estructuras**: devuelve en una lista de las distintas estructuras que se encuen-

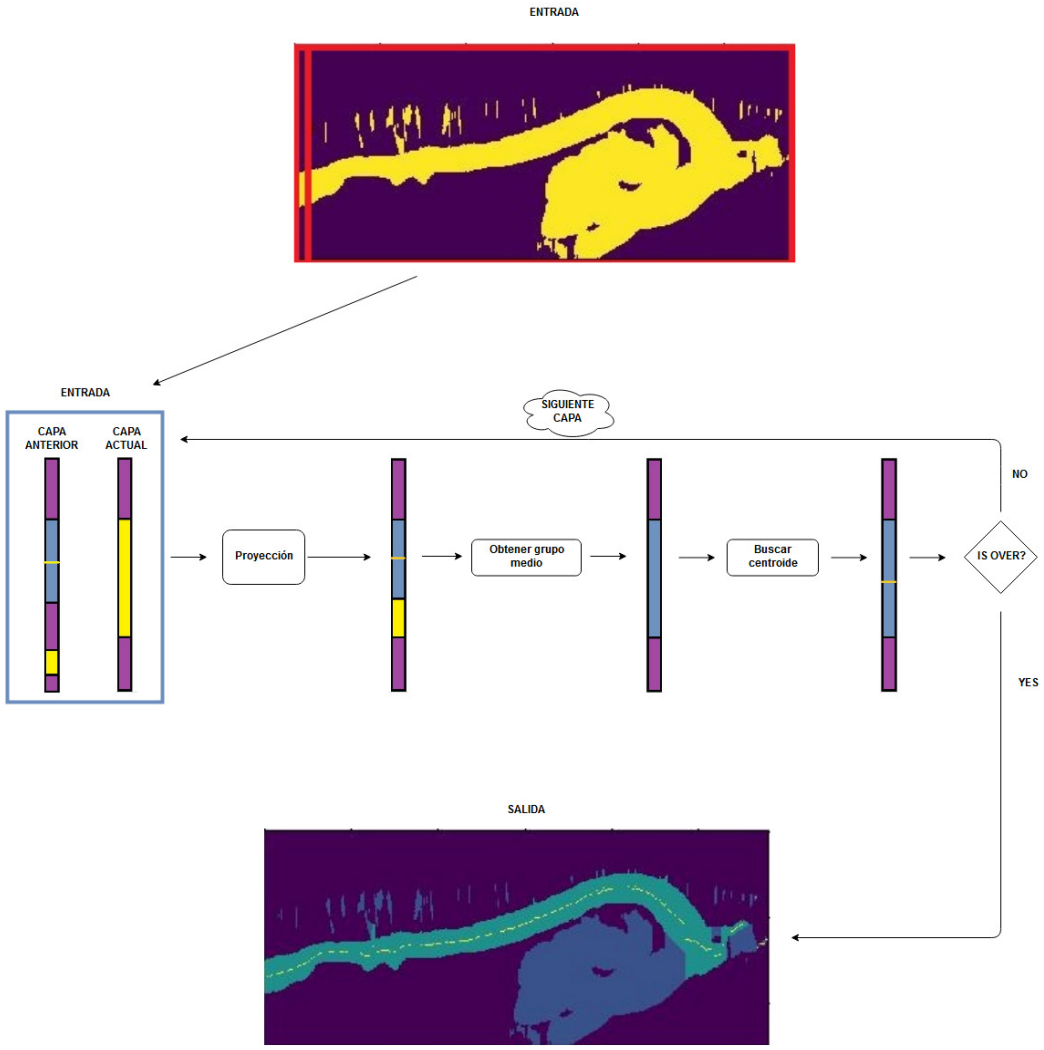


Figura 5.11: **Seguimiento de forma 2D**: procesado del resto de capas.

tran en la capa seleccionada con el siguiente formato:
(índice punto inicial, área estructura).

2. **Obtener estructura principal**: Se obtiene cual es la estructura con mayor área detectada en el paso anterior dando por hecho que dicha estructura es la aorta.

La precondición de este método es que en la capa inicial debe haber al menos una estructura y que la estructura más grande debe ser la estructura principal. Esto conlleva dos posibles riesgos: que la capa este vacía o que haya más de una. Sin embargo, es poco probable dado que la UNet es muy eficiente en este tipo de capas y el algoritmo es capaz de recuperarse en la siguiente capa si fuera necesario.
Operar el resto de capas ya no se basa en que la estructura más grande forme parte de la aorta, sino en la estructura que ocupa la posición más cercana con respecto a la capa anterior. Las fases de esta etapa del algoritmo son las siguientes (Figura 5.11):

1. **Calcular el centroide**: Recibe la estructura que forma la aorta de la capa anterior y devuelve el centroide.

2. **Proyectar estructura en la capa superior** siguiendo el siguiente algoritmo:

$$P_k(i) = \begin{cases} n & Si \quad P_k(i) = n \quad \cap \quad P_{k-1}(i) = 1, \\ m & Si \quad P_k(i) = m \quad \cap \quad P_{k-1}(i) = 1, \\ P_{k-1}(i) & resto \end{cases} \tag{5.16}$$

   siendo n el valor del píxel si pertenece a la estructura seleccionada de la capa anterior, m el valor de su píxel centroide e i la coordenada del vector P_k.

3. **Obtener estructuras intermedias**: Se obtienen todas las estructuras conectadas en la capa actual y se selecciona aquella cuyo centroide esté más próximo al centroide de la aorta de la capa anterior.

4. **Reformar estructura**: Se eliminan partes de la estructura con el fin de mantener una proporción similar a lo de la capa anterior.

El algoritmo funciona en la gran mayoría de los casos. Sin embargo es necesario colocar reglas de recuperación para determinados casos en los que falla:

- Si una capa no tiene ninguna estructura que contenga el punto centroide de la aorta en la capa anterior, se busca la estructura más cercana a ese punto y se toma como forma de la aorta.

- Si no se encuentra ninguna estructura cercana o la capa está completamente vacía, se sube el centroide de la capa anterior para que el algoritmo pueda continuar en la siguiente. Esta regla solo se puede aplicar 3 veces seguidas. Si se sobrepasa ese número se borran las 3 últimas capas y el algoritmo finaliza.

En la Figura 5.12 se muestran los resultados del algoritmo. En verde se representa la aorta, en azul, el ruido eliminado y en amarillo, el centroide de cada capa. Por otra parte, podemos ver como quedaría la conversión 3D en la Figura 5.13.

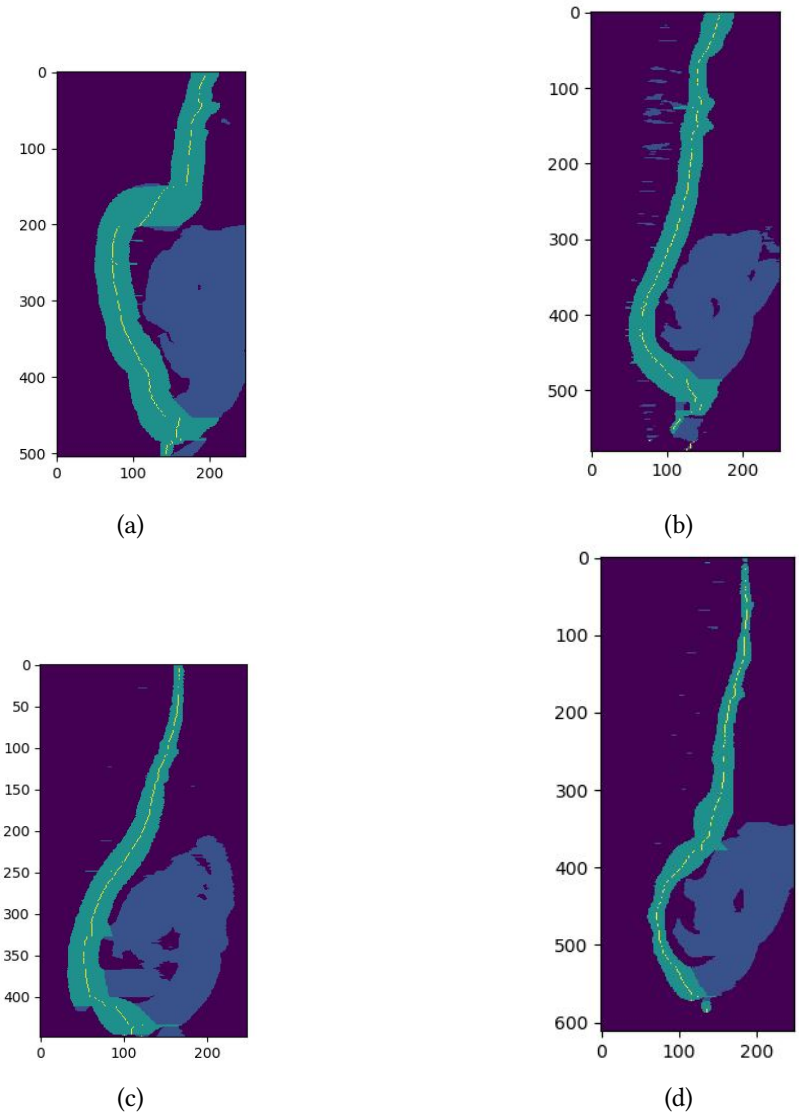
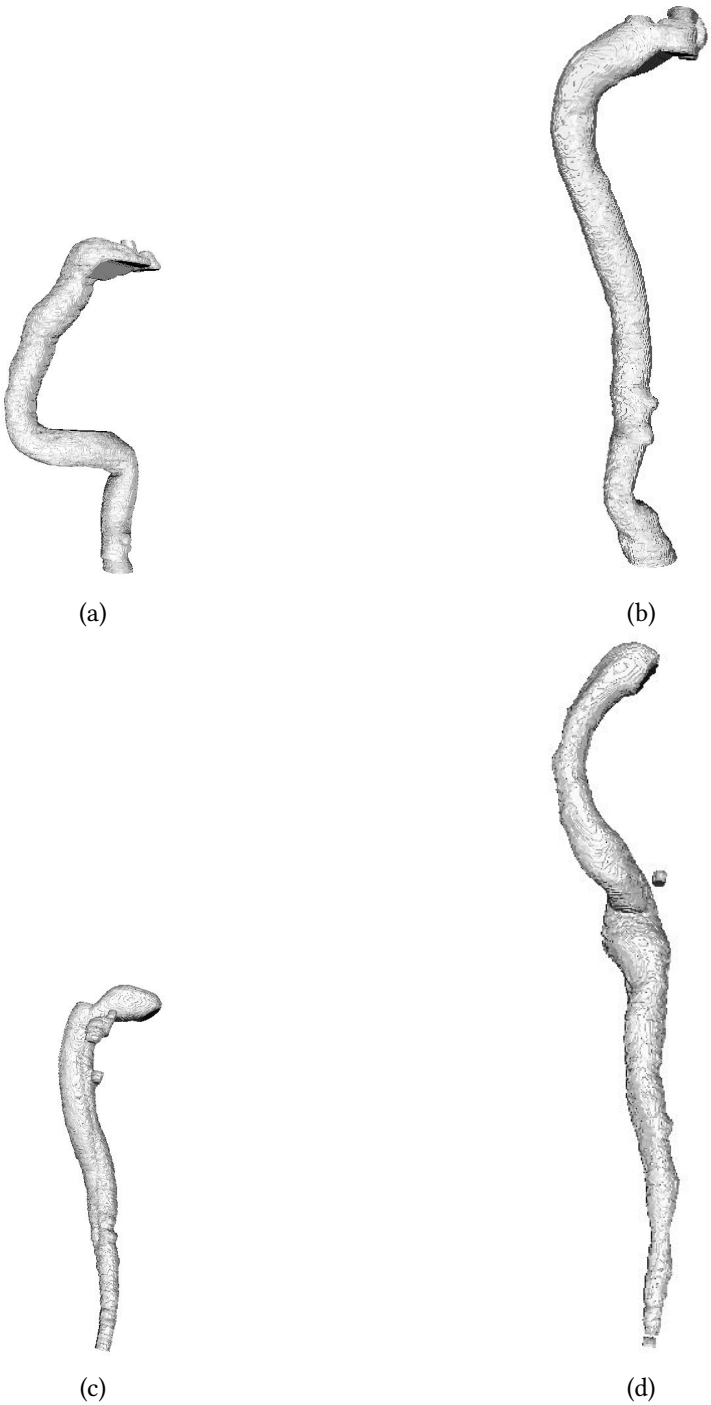


Figura 5.12: **Resultados obtenidos en el método seguimiento de forma 2D**: a) Caso 119. b) Caso 139. c) Caso 173 . d) Caso 164.



(a) (b)

(c) (d)

Figura 5.13: **Resultado final del método de seguimiento de forma 2D convertido a 3D**: a) Caso 119. b) Caso 139. c) Caso 173. d) Caso 164.

5.5 Segmentación a partir del arco aórtico

Los métodos de postprocesado anteriores se ocupan de procesar la parte descendente de la aorta y del arco aórtico. Se diseñaron de tal manera por las dificultades técnicas que requiere separar la aorta ascendente del corazón. Esto es debido a su proximidad y similaridad de forma cuanto más nos acercamos a la entrada de este órgano. Los métodos de segmentación a partir del arco aórtico se ocuparan de realizar un seguimiento hacia abajo por la aorta ascendente.

Los métodos de segmentación a partir del arco aórtico no pueden ser de *seguimiento en 2D a partir de PCA* dado a la superposición de estructuras de la aorta y el corazón. Pero si pueden sacar ventaja a los resultados obtenidos en los algoritmos de seguimiento 2D para segmentar la parte de la aorta ascendente y el corazón y, de esta forma, poder reducir considerablemente el número de datos que se tienen que procesar (Figura 5.14). Una vez realizado el método de segmentación a partir del arco aórtico se unen ambos resultados obtenidos en los distintos seguimientos para disponer de la segmentación final (Figura 5.15).

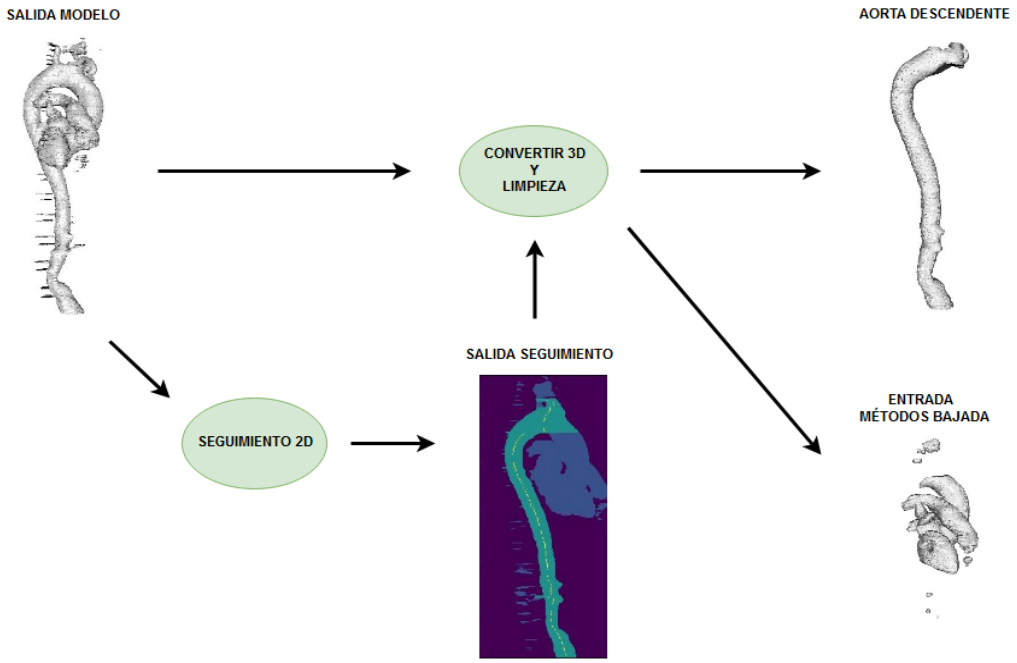


Figura 5.14: **Datos entrada métodos de segmentación a partir del arco aórtico**

5.5.1 Proyección y seguimiento de puntos a partir del arco aórtico

Este método es una adaptación de *Proyección y seguimiento de puntos*. En el seguimiento de la aorta descendente métodos de este estilo son muy lentos pero ahora al disponer de un

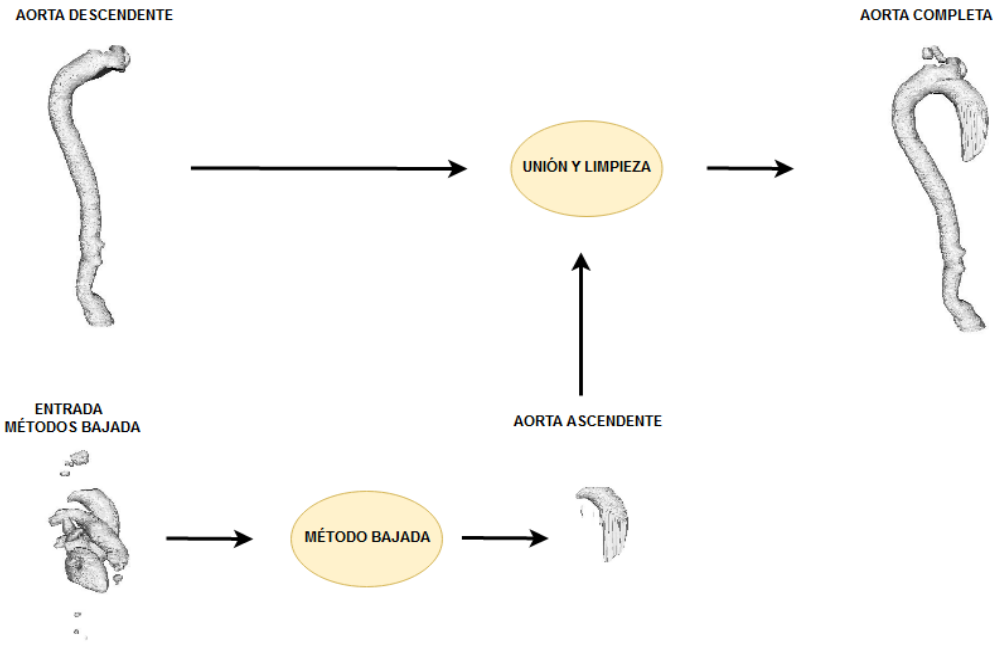


Figura 5.15: **Salida de los métodos de segmentación a partir del arco aórtico**

número tan reducido de datos los tiempos de ejecución son aceptables. Este método se divide en tres partes:

1. **Detectar inicio de la aorta**: Encontrar donde empieza la aorta para iniciar el proceso de seguimiento.

2. **Realizar seguimiento** de la aorta ascendente.

3. **Detectar fin de la aorta**: Encontrar el punto final donde acaba la aorta. Hay dos formas de hacerlo:

    - Controlando la densidad de puntos en cada capa y parándolo cuando llegue a un valor determinado.
    - Cuando llega al final del corazón se remata la operación. Se considera que ha llegado a ese punto cuando el algoritmo detecta varias capas vacías seguidas.

Las distintas partes de la aorta utilizadas en este algoritmo se pueden ver en la Figura 5.16. El seguimiento se compone así de tres pasos:

1. **Proyección capa superior**: Se proyecta la capa superior a la capa actual siguiendo la

siguiente ecuación:

$$P_k(i,j) = \begin{cases} n & Si \quad I_k(i,j) \cap P_{k+1}(i,j) \neq 0, \\ P_{k-1}(i,j) & resto \end{cases} \tag{5.17}$$

donde n es es valor que queremos asignar en la proyección e (i, j) son las coordenadas de la imagen I_k y P_{k+1} representa los puntos proyectados.

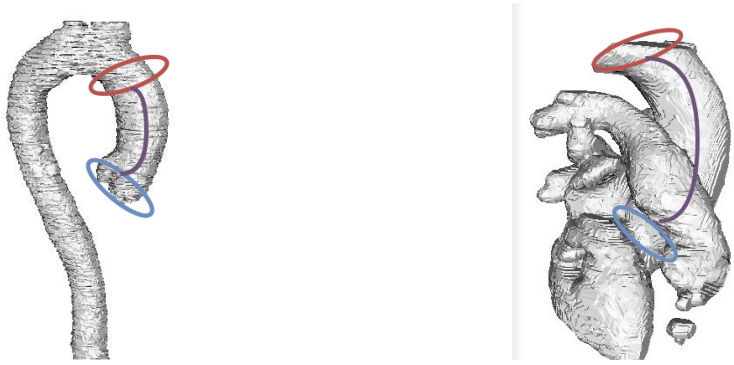
2. **Desplazamiento capa actual**: Siendo O_k conjunto de puntos no cero de la capa k y P_k los proyectados se seleccionan los puntos que devuelve la siguiente ecuación:

$$D_k = O_k - P_k \tag{5.18}$$

3. **Seguimiento de puntos**: Busca los puntos del conjunto D_k que están unidos a algún punto del conjunto P_k (Figura 5.4).

$$S_k(i,j) = \begin{cases} 1 & Si \quad P_k(i,j) = 1 \cup D_k(l,m) = 1 \cap d(l,m) \in N(i,j), \\ 0 & resto \end{cases} \tag{5.19}$$

donde $N(i, j)$ es el conjunto de 8 píxeles vecinos del píxel (i, j). Sin embargo, el número de píxeles que pueden ser 1 están limitados para mantener la proporción de la capa anterior y así evitar que la segmentación obtenga partes del corazón.



(a) (b)

Figura 5.16: **Partes del método de seguimiento**: a) Partes en la aorta completa. b) Partes en la entrada del método de métodos de segmentación a partir del arco aórtico.

Capítulo 6

Resultados

En este capítulo se comentarán de los distintos resultados obtenidos en el proyecto. También se explicará su interpretación y lo que suponen a la hora de segmentar una aorta.

6.1 Resultados de las primeras generaciones de UNet

Los resultados obtenidos en las primeras generaciones muestran un alto nivel de exactitud, pero con valores altos en la función *loss* (Tabla 6.1). Es por ello por lo que se pueden considerar malos resultados. Normalmente ambos valores son dependientes entre sí, si uno aumenta el otro disminuye y viceversa, que no se cumpla esta norma es una indicación de que las métricas usadas no son las correctas. El $modelo\_3.a$ no tiene valores dado que fueron tan malos que se consideraron nulos. Cabe destacar que los modelos presentados aquí no son el total de modelos entrenados sino los que fueron más representativos para cada generación.

Tabla 6.1: **Resultados de las primeras cinco generaciones.**

Nombre modelo	Precisión	Función Loss
modelo_1.a	0.87	-51.42
modelo_2.a	0.90	-95.66
modelo_3.a	––	––
modelo_4.a	0.92	-42.45
modelo_5.a	0.90	-49.88
modelo_5.b	0.91	-35.21
modelo_5.c	0.90	-43.05
modelo_5.d	0.90	-25
modelo_5.e	0.90	-18.31

6.2 Resultados de las últimas generaciones de UNet

Este apartado se ocupa de explicar los resultados de las generaciones seis, siete y ocho (Tabla 6.2). Todas ellas tienen en común en que se utilizó el *Coeficiente Sørensen–Dice (DSC)* como métrica para medir su precisión.

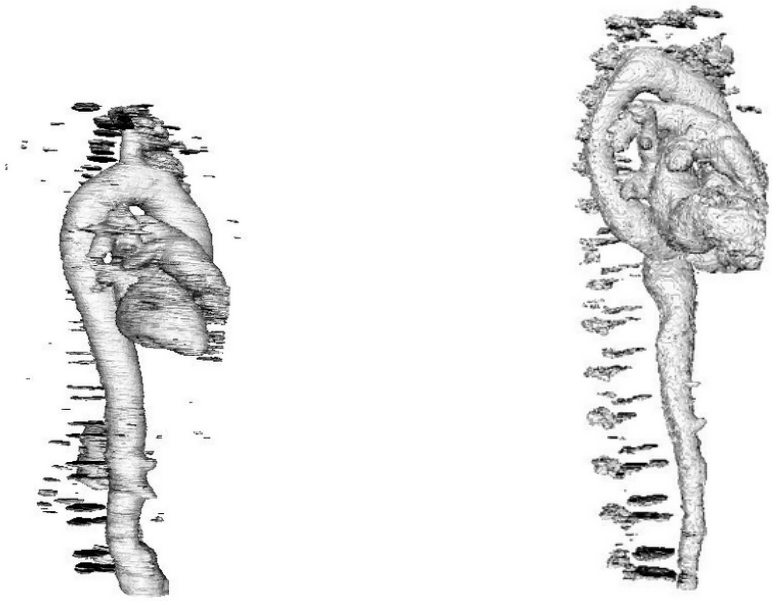
- En la **sexta generación** se testearon tantó esta métrica como distintas funciones *loss* para obtener el mejor resultado. La única que dio resultados óptimos fue la de *Jaccard*, los modelos con nomenclatura $modelo\_6.3.*$. Como se puede observar en la Figura 6.1 la aorta es visible pero la segmentación obtiene varias estructuras alrededor que no forman parte de ella. Estas estructuras principalmente son fragmentos de huesos, partes del corazón e incluso riñones, partes del organismo que comparten forma y niveles de gris similares a los de la aorta.

- En la **séptima generación** se intentó solucionar estos problemas con el preprocesado de *Imagen con profundidad* pero, a pesar de obtener resultados similares en la fase de entrenamiento que la sexta generación, las segmentaciones son peores. Esto se debe a que los falsos positivos en una capa no solo son de esa capa, sino de todas las capas que se unieron en una en el preprocesado (Figura 6.6).

- La **octava generación** fue la mejor de todas al conseguir eliminar gran parte de los falsos positivos obtenidos en la sexta generación (Figura 6.3). El modelo final seleccionado para ser utilizado en la segmentación de la aorta fue el $modelo\_8.c$ (Figura 6.4).

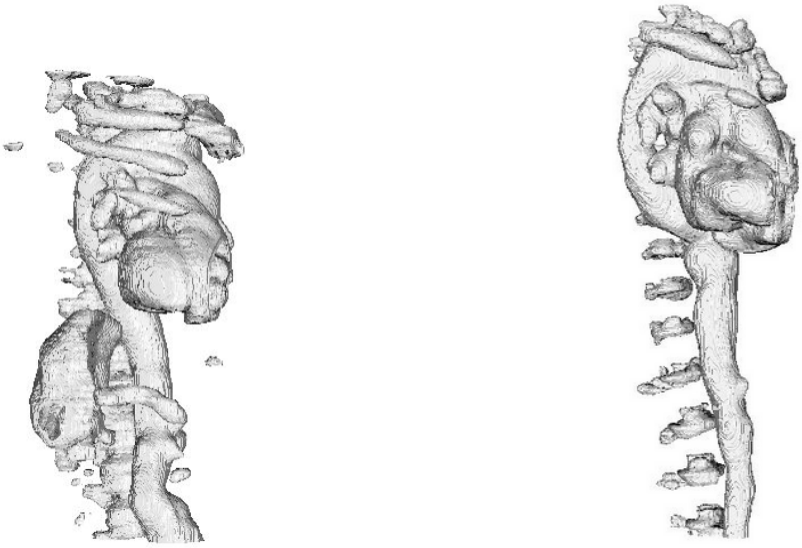
Tabla 6.2: **Resultados de las generaciones restantes.**

Nombre modelos	**Nº epochs**	**DSC**	**Loss**
modelo_6.1	1	1.96	200
modelo_6.2	1	1.96	200
modelo_6.3.a	1	0.89	-8.74
modelo_6.3.b	1	0.78	-8.62
modelo_6.3.c	1	0.82	-9.37
modelo_7.a	5	0.88	-9.25
modelo_7.b	5	0.85	-8.78
modelo_7.c	5	0.86	-8.91
modelo_8.a	1	0.95	-6.35
modelo_8.b	1	0.95	-6.86
modelo_8.c	1	0.96	-6.44



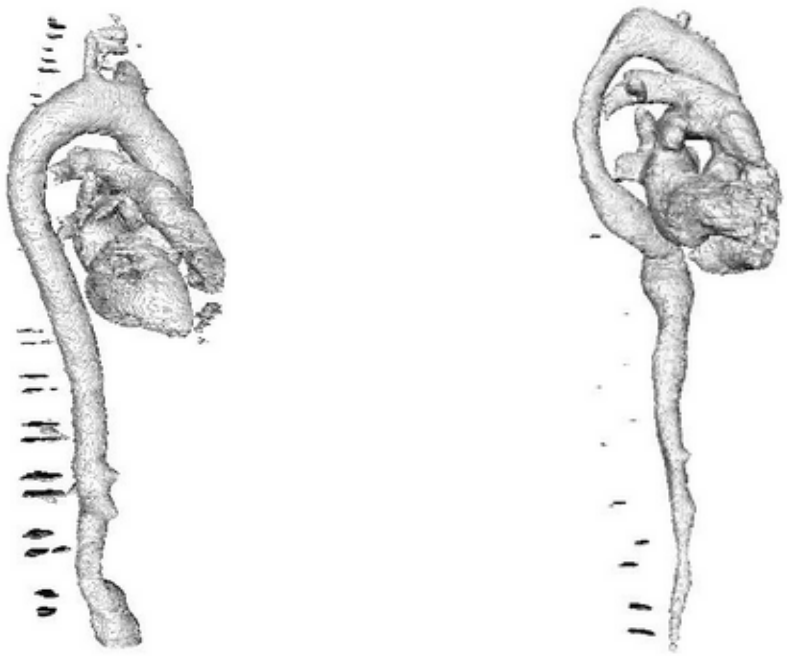
(a) (b)

Figura 6.1: **Segmentaciones obtenidas de los modelos de la sexta generación**: a) Caso 139. b) Caso 164.



(a) (b)

Figura 6.2: **Segmentaciones obtenidas de los modelos de la septima generación**: a) Caso 139. b) Caso 164.



(a) (b)

Figura 6.3: **Segmentaciones obtenidas de los modelos de la octava generación**: a) Caso 139. b) Caso 164.

6.3 Segmentación automática aorta hasta el arco aórtico

En este apartado se testeó la precisión y el tiempo necesario para la segmentaron de un conjunto de casos de la base de datos. Todo el proceso fue realizado en en el entorno de trabajo *Colab research* con optimización de GPU. Los métodos seleccionados para el testeo son los siguientes:

- El método *Imagen original filtrada* para el preprocesado.
- El modelo $modelo\_8.c$ para el modelo.
- El método de *seguimiento de forma 2D* para el postprocesado.

Cada uno de los métodos seleccionados se escogieron al ser los mejores en cuestiones de precisión y tiempo de ejecución en cada una de las partes. En la Figura 6.5 se puede observar el esquema final del algoritmo de segmentación.
En la Tabla 6.3 se pueden ver la precisión de la segmentación en los distintos casos dando buenos resultados. Los tiempos de ejecución son buenos al no pasar de los quince segundos en los resultados (Tabla 6.4)

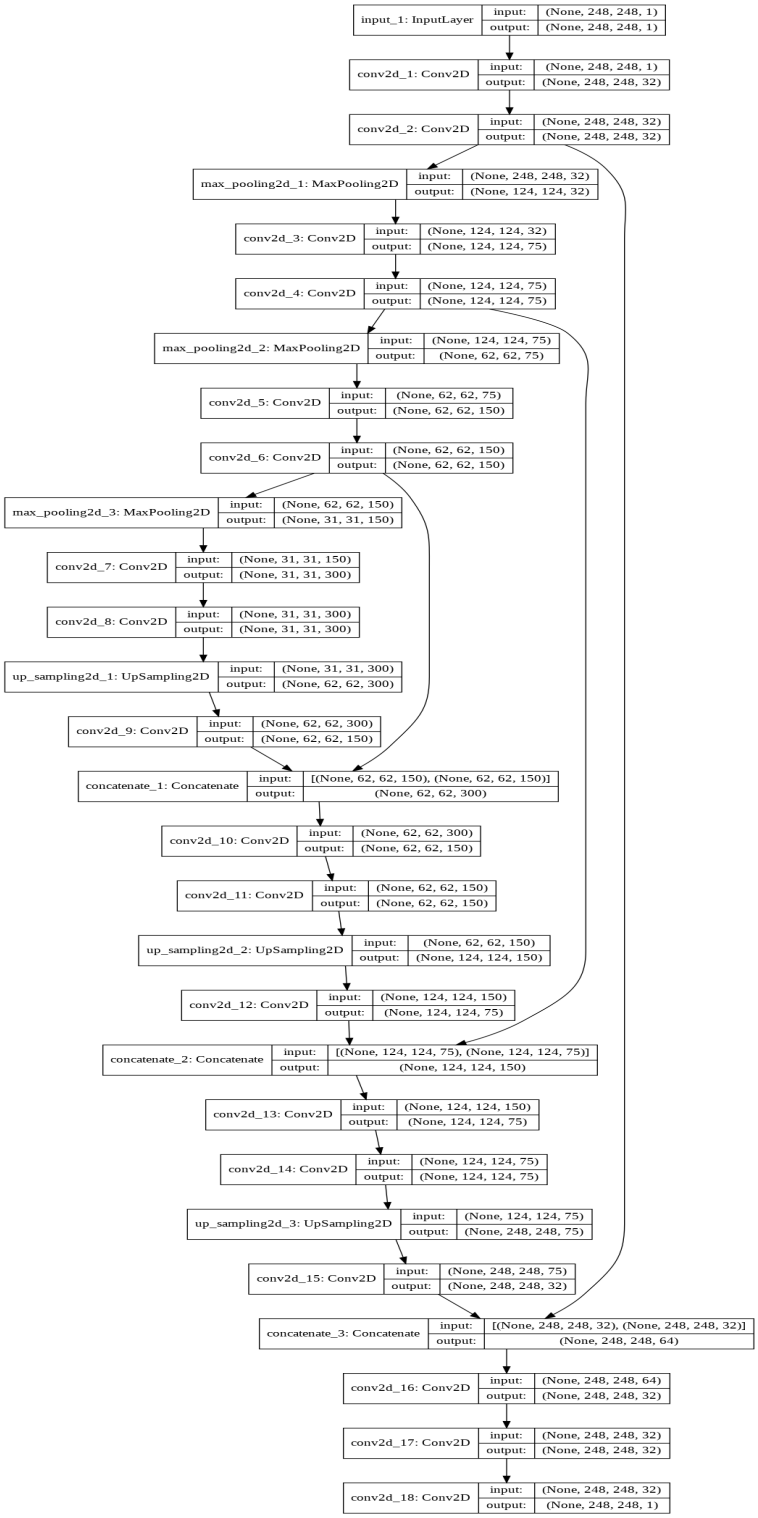


Figura 6.4: **Modelo final.**

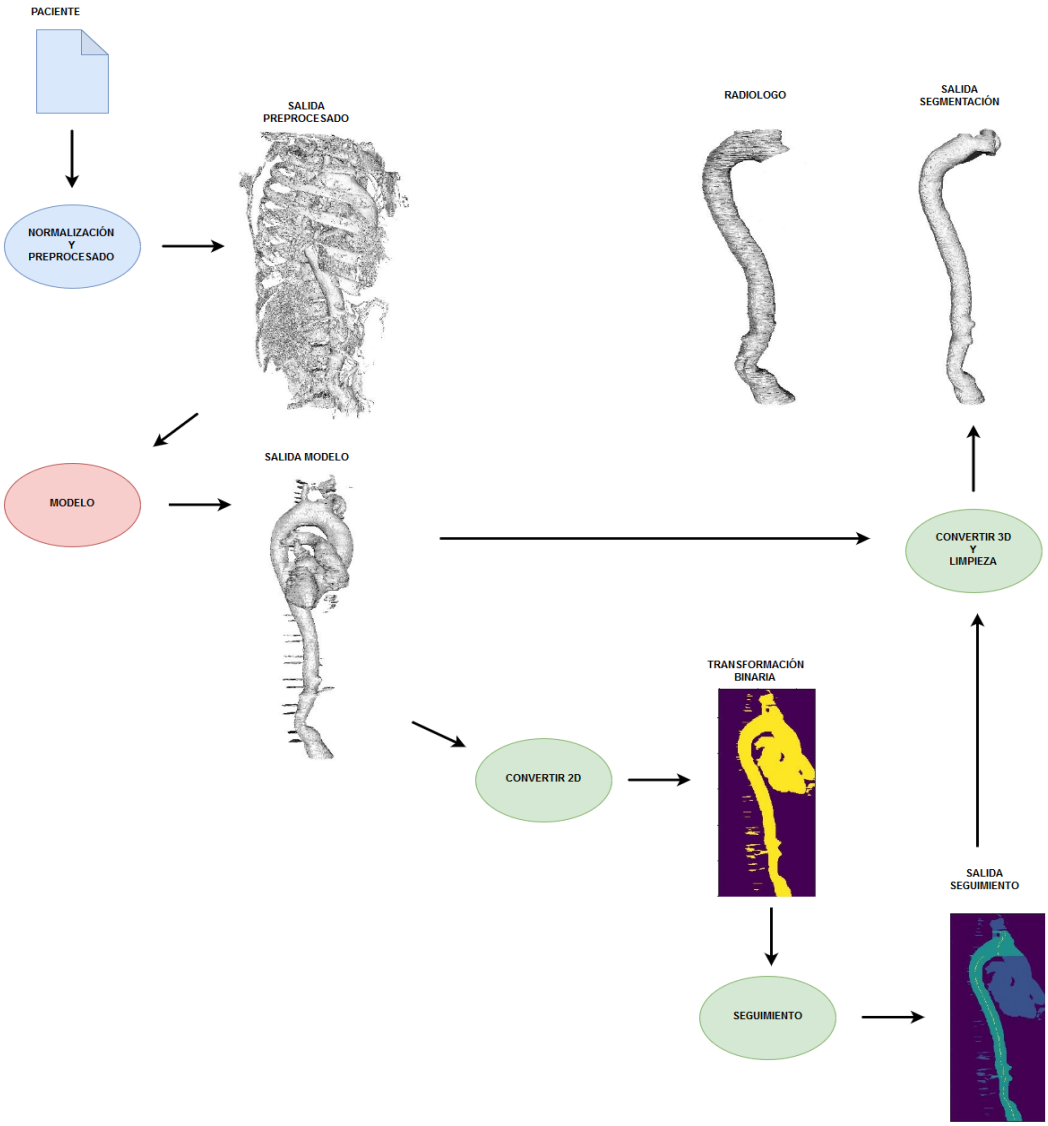


Figura 6.5: **Algoritmo usado para la segmentación de la aorta hasta el arco aórtico.**

Tabla 6.3: **Resultados segmentación de casos de la aorta hasta el arco aórtico.**

Número caso	DSC
53	0.84
55	0.89
119	0.92
139	0.89
164	0.86
173	0.85

Tabla 6.4: **Tiempos obtenidos en el proceso de segmentación.**

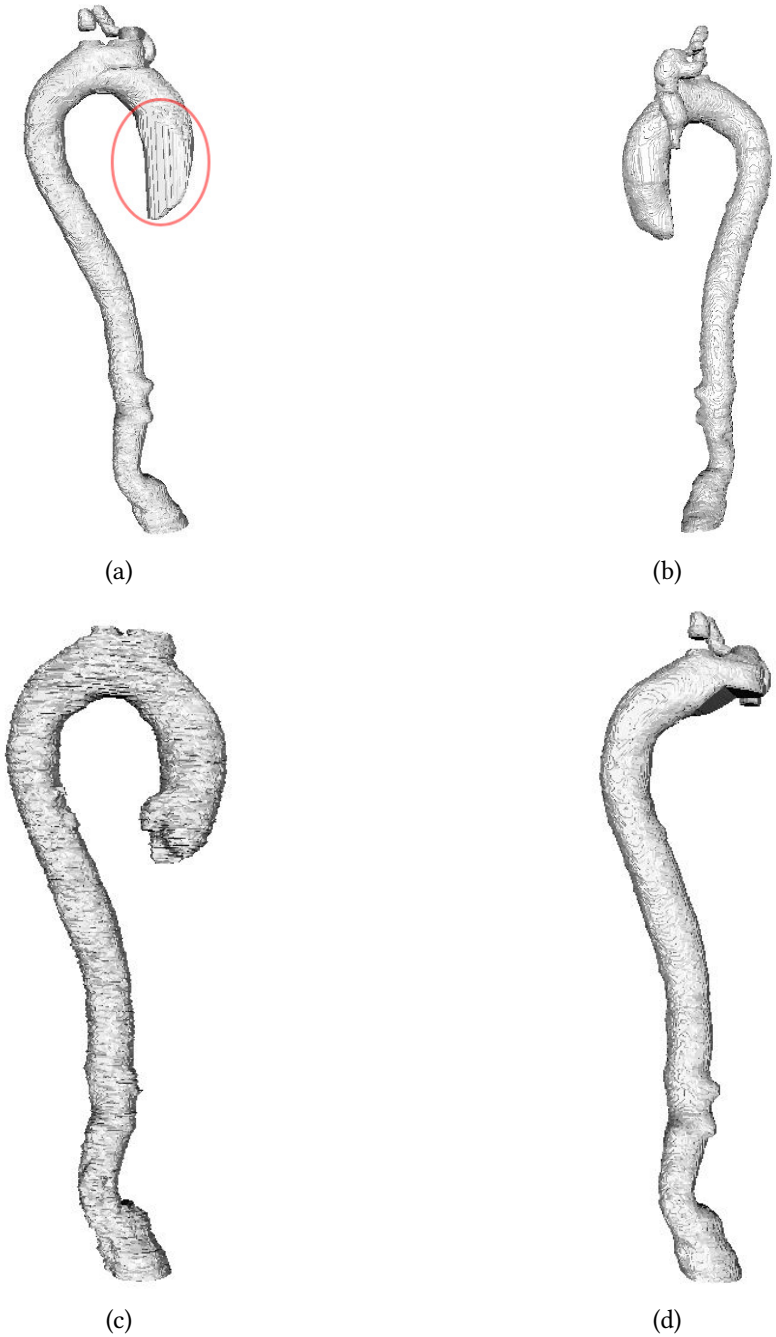
Número caso	Preprocesado	Modelo	Postprocesado	Tiempo total
53	0.10s	5.03s	0.75s	5.88s
55	0.48s	6.61s	3.36s	10.70s
119	0.77s	6.42s	3.58s	10.77s
139	0.86s	7.05s	4.10s	12.58s
164	0.89s	7.00s	4.35s	12.28s
173	0.68s	6.68s	3.10s	10.47s

6.4 Segmentación semiautomática aorta completa

Para este testeo se seleccionaron las segmentaciones obtenidas en el apartado anterior (Figura 6.6d) y se les aplicó el *método de segmentación a partir del arco aórtico*. Se requiere por cada caso que se regulen dos parámetros manualmente para referenciar donde se inicia el proceso. Como se puede ver la mayoría de resultados son buenos rondando entre el 87-90 por ciento (Tabla 6.5). Sin embargo, el método da ciertos problemas en la entrada del corazón, al separar la aorta de este órgano se genera un corte artificial (Figura 6.6a). Este corte solo se da en una proyección, quedando invisible desde cualquier otro ángulo de la aorta (Figura 6.6b). Los tiempos de procesado están en los dos segundos de media.

Tabla 6.5: **Resultados segmentación de casos de la aorta completa.**

Número caso	DSC
53	0.76
55	0.87
119	0.92
139	0.90
164	0.87
173	0.87



(a) (b)

(c) (d)

Figura 6.6: **Segmentaciones obtenidas de la aorta completa**: a, b) Segmentación realizada desde distintas proyecciones.c) Segmentación realizada por el radiólogo. d) Entrada del algoritmo

Capítulo 7

Aplicación de escritorio

En este capítulo se explicará el diseño y desarrollo de la aplicación de escritorio para el segmentado y visualización de la aorta. Se describirán los mockups, casos de uso, pruebas y la versión final obtenida el proyecto.

La aplicación se diseñó para su uso en investigación de cara a poder testear el algoritmo de segmentación y su aplicación en futuros proyectos. De esta manera debería poder permitir un uso general del mismo y permitir con facilidad añadir o eliminar funcionalidades para poder adaptarla a los requisitos requeridos.

7.1 Análisis

El análisis se dividió en dos partes. La parte inicial, donde se estudió las funcionalidades que debería cumplir la aplicación y la segunda, donde se diseñó la interfaz gráfica de tal manera que sea sencilla de usar y entender por el usuario.

7.1.1 Casos de uso

La aplicación de escritorio debe cumplir dos tareas, realizar la segmentación de un caso clínico y poder visualizarlo. Los siguientes diagramas de casos de uso detallan las funcionalidades de la aplicación:

- **Diagrama de casos de uso de la segmentación del caso clínico** (Figura 7.1), visualiza las distintas partes del algoritmo creado en el proyecto y como interactúan entre sí (Tabla 7.1).
- **Diagrama de casos de uso de la visualización del caso clínico** (Figura 7.2), para poder analizar mejor la aorta es necesario disponer de una serie de herramientas que nos permitan interactuar con ella. Por ello se añadieron funcionalidades para poder rotar la figura, aplicarle una máscara para facilitar comparaciones, poder recortarla en

distintos tamaños (Tabla 7.5), poder realizar una conversión a 3D (Tabla 7.4), cambiar la vista (Tabla 7.6), poder abrir otro caso (Tabla 7.3) o guardar el caso actual (Tabla 7.2).

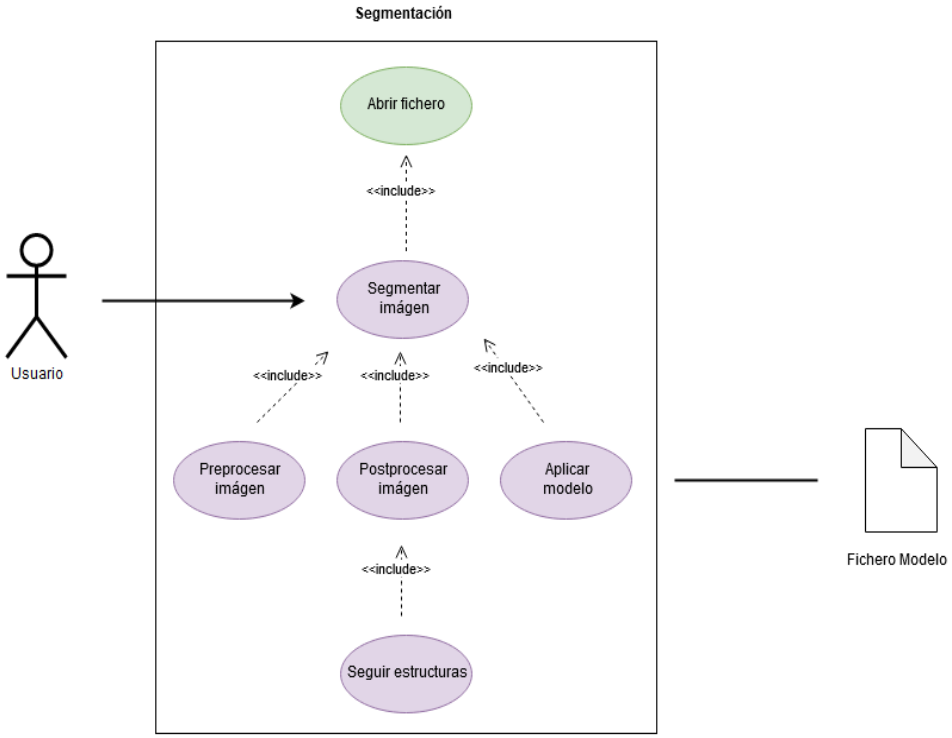


Figura 7.1: **Diagrama de casos de uso de la segmentación del caso clínico.**

7.1.2 Interfaz gráfica

En el diseño de la interfaz se tomó la decisión de que el programa pudiera abrir varios casos de manera simultánea con el fin de poder simplificar el análisis de los mismos. Por ello se diseñaron dos tipos de ventanas para poder cumplir este fin:

- **Ventana principal** (Figura 7.3), ventana desde donde se abren los distintos casos y se decide si se segmentan o no. Una vez realizada esta operación el programa abrirá una ventana secundaria que lo visualice. Es importante que permita abrir distintos tipos de formatos para poder generalizar su uso (Casos de uso: SEG001, AUX002).

- **Ventanas secundarias**, ventanas encargadas de la visualización de un caso. Se encargan de facilitar al usuario todas las funcionalidades propuestas en el diagrama de casos de uso de la visualización del caso clínico (Casos de uso: AUX001, AUX003, VIS001, VIS002). Existen dos tipos de ventanas:

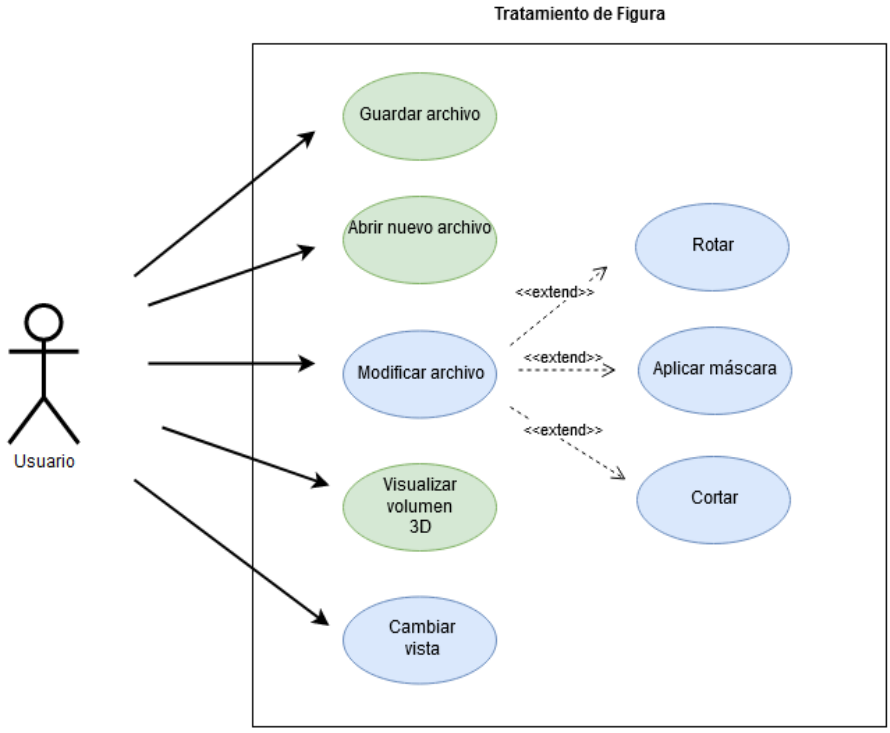


Figura 7.2: **Diagrama de casos de usos de la visualización del caso clínico.**

Tabla 7.1: Caso de uso segmentar imagen

SEG001	**Segmentar imagen**
Descripción	Realiza la segmentación de un caso seleccionado por el usuario.
Precondición	Las proporciones del caso seleccionado por el usuario son correctas.
Secuencia normal	• El usuario solicita una segmentación sobre un caso clínico. • El sistema realiza el preprocesado sobre el caso. • El sistema utiliza el modelo para realizar la segmentación. • El sistema realiza el postprocesado sobre el caso.
Postcondición	Devuelve el caso segmentado.
Excepciones	Formato incorrecto.
Importancia	Alta

Tabla 7.2: Caso de uso guardar archivo

AUX001	**Guardar archivo**
Descripción	Guarda el caso en la memoria de un directorio determinado.
Precondición	Debe haber un caso cargado en la memoria del programa.
Secuencia normal	• El usuario solicita guardar un archivo. • El usuario inserta la ruta y el formato. • El sistema guarda el archivo.
Postcondición	Realiza la escritura del caso en un formato determinado.
Excepciones	
Importancia	Media

Tabla 7.3: Caso de uso abrir nuevo archivo

AUX002	**Abrir nuevo archivo**
Descripción	Cargar un caso para poder visualizarlo.
Precondición	El path debe existir y el formato debe existir en la aplicación.
Secuencia normal	• El usuario solicita abrir un archivo. • El usuario inserta la ruta y el formato. • El sistema abre el archivo.
Postcondición	Carga el fichero en el programa.
Excepciones	Formato incorrecto
Importancia	Media

Tabla 7.4: Caso de uso realizar conversión

AUX003	**Visualizar volumen 3D**
Descripción	Renderiza y visualiza el caso en 3D.
Precondición	Debe haber un caso cargado en la memoria del programa.
Secuencia normal	• El usuario solicita el renderizado. • El sistema realiza el renderizado.
Postcondición	Devuelve una representación 3D del caso.
Excepciones	Figura no compatible.
Importancia	Baja

Tabla 7.5: Caso de uso modificar archivo

VIS001	**Modificar archivo**
Descripción	Editar un caso según las preferencias del usuario.
Precondición	Debe haber un caso cargado en la memoria del programa.
Secuencia normal	• El usuario solicita modificar un archivo. • El usuario inserta el tipo de modificación. • El sistema realiza las modificaciones en el archivo.
Postcondición	Devuelve el caso modificado.
Excepciones	
Importancia	Baja

Tabla 7.6: Caso de uso cambiar vista

VIS002	**Cambiar vista**
Descripción	Modifica la ventana de visualización.
Precondición	Debe haber uno o dos casos cargados en memoria
Secuencia normal	• El usuario solicita un cambio de vista. • El sistema realiza el cambio.
Postcondición	Cambia la ventana de la visualización.
Excepciones	Formato incorrecto.
Importancia	Baja

– **Ventanas secundarias simples** (Figura 7.4), visualizan cada uno de los cortes del caso. Su diseño se basa en una imagen grande del corte seleccionado y una lista de todos los cortes a la izquierda para agilizar su navegación. En la parte inferior de la ventana aparecerán los botones con todas las funcionalidades correspondientes.

– **Ventanas secundarias dobles** (Figura 7.5), muy similares a las ventanas simples con la excepción de que hay dos imágenes grandes en vez de una. Esto permitirá observar caso segmentado y sin segmentar al mismo tiempo entre otro tipo de comparaciones.

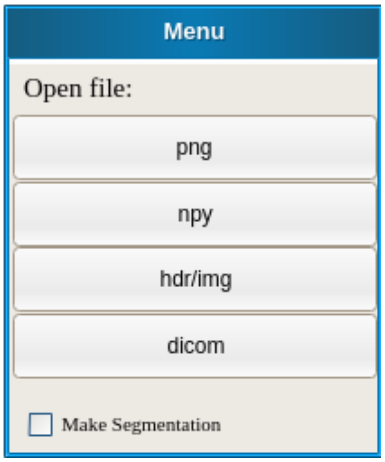


Figura 7.3: **Mockup aplicación escritorio pantalla inicial.**

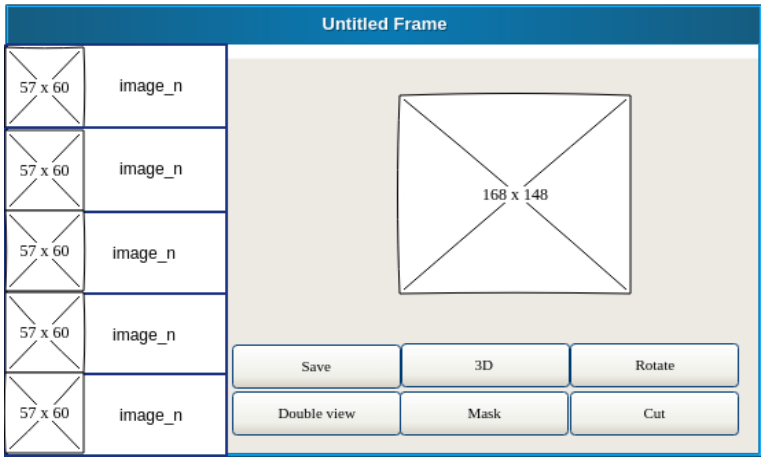


Figura 7.4: **Mockup aplicación escritorio ventana secundaria simple.**

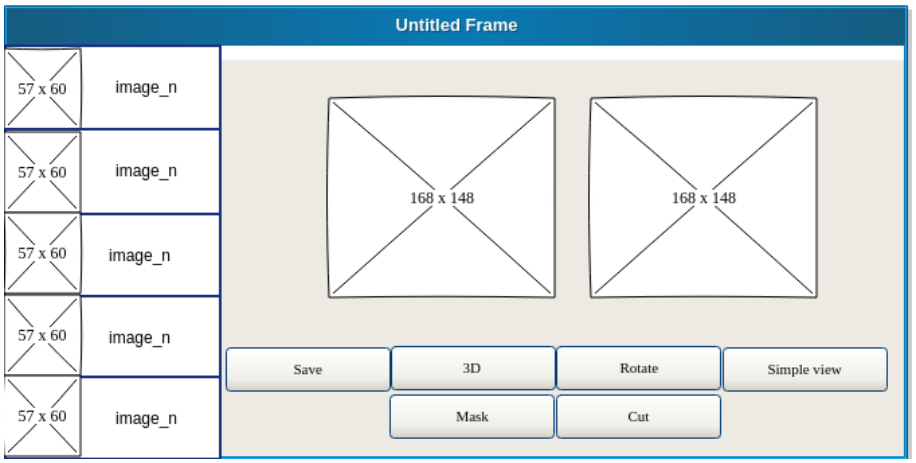


Figura 7.5: **Mockup aplicación escritorio ventana secundaria doble.**

7.2 Diseño

En la Figura 7.6 se pueden observar las distintas clases que componen la aplicación asociadas a la ventana principal de la aplicación. Desde ella se van generando las distintas ventanas secundarias (Figura 7.8). Cada una comparte un patrón modelo, vista, controlador (MVC) siendo cada una de las pantallas secundarias independientes entre si.
El algoritmo de segmentación está dividido en una serie de clases y librerías (Figura 7.7) encargadas de la segmentación del caso seleccionado. Las clases auxiliares se dividen en dos clases, la primera encargada del tratamiento de los distintos formatos del proyecto y la segunda de las visualizaciones en 3D (Figura 7.9). Las ventanas diálogo son ventanas extras que se generan en las ventanas secundarias, todas ellas tienen su propio controlador y interfaz pero comparten el mismo modelo de la ventana secundaria.

7.3 Implementación

Por lo que respecta a la implementación de la aplicación, ésta se inicia desde la ventana principal (Figura 7.11), donde se pueden abrir varios casos para su visualización o segmentación en distintos formatos. La selección de archivos tanto para ser abiertos como para ser salvados se hace a partir de la ventana diálogo estándar de GTK (Figura 7.12). La visualización de los distintos casos o resultados de la segmentación se hace a partir de las pantallas secundarias (Figuras 7.13, 7.14). Desde ellas se puede ir examinando los distintos cortes, rotar el caso sobre sus ejes, recortarlo (Figura 7.15), realizar un renderizado 3D (Figura 7.16) y guardarlo en otra proyección. A la derecha de la ventana se puede ver una lista con todos los cortes en miniatura para facilitar la navegación de los mismos. En la Figura 7.17 se puede ver

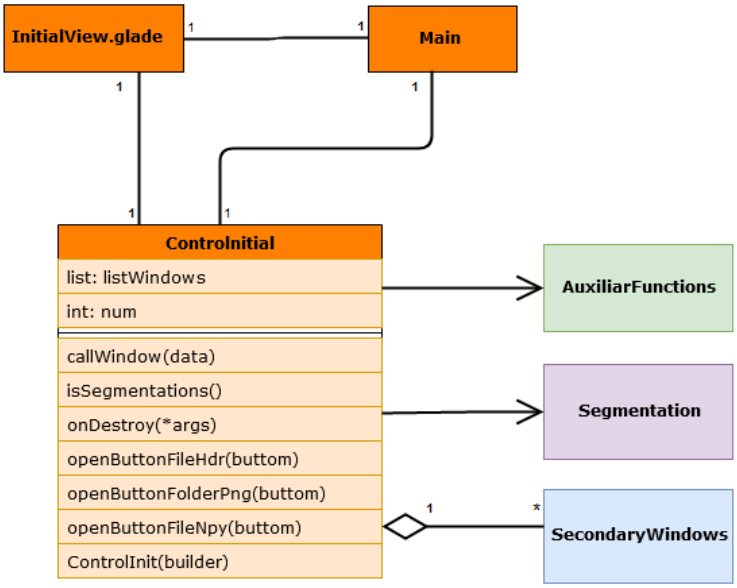


Figura 7.6: **Diagrama de clases de la aplicación.**

una captura de pantalla de la interfaz completa.

7.4 Pruebas

Para comprobar el correcto funcionamiento de la aplicación se hicieron pruebas de caja negra para testear sus distintas funcionalidades (Tabla 7.8). Cada prueba tiene un código único que viene determinado por el tipo de prueba realizada. Siendo las pruebas que comienzan por E pruebas de los datos de entrada, S pruebas de salida de datos y M pruebas de modificación de datos en plena ejecución. Los resultados esperados por cada una de las pruebas están recogidas en la siguiente Tabla 7.7. La aplicación pasó todas las pruebas descritas en este apartado.

Tabla 7.7: **Resultados esperados de pruebas de caja negra**

Codigo	Nombre	Descripción
EC	Error controlado.	Salta un error controlado.
FC	Funciona correctamente.	El programa funciona correctamente.
NP	No permitido.	Acción que no debería permitir el programa.

7.5 Instalación de la aplicación

La aplicación se ha probado en distintas distribuciones Linux. Sin embargo su uso debería ser compatible para *OS* y *Windows*. Para la instalación de las dependencias se usó *miniconda*

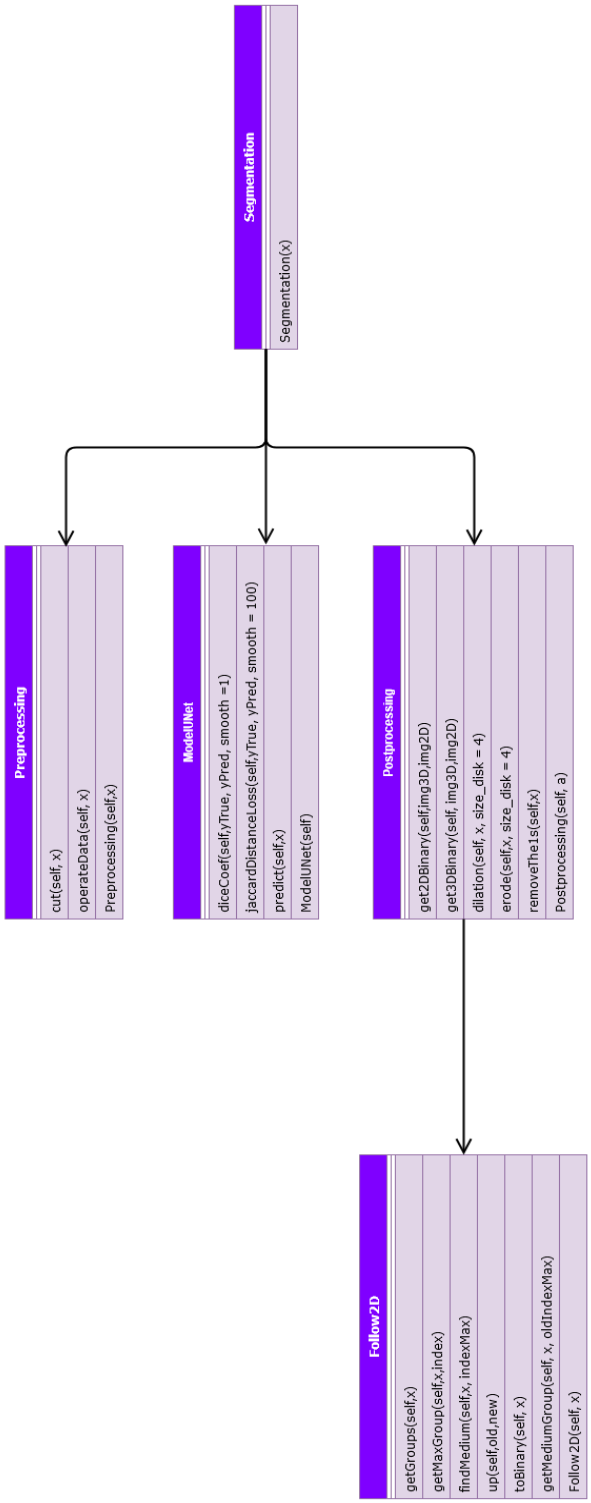


Figura 7.7: **Diagrama de clases de la segmentación.**

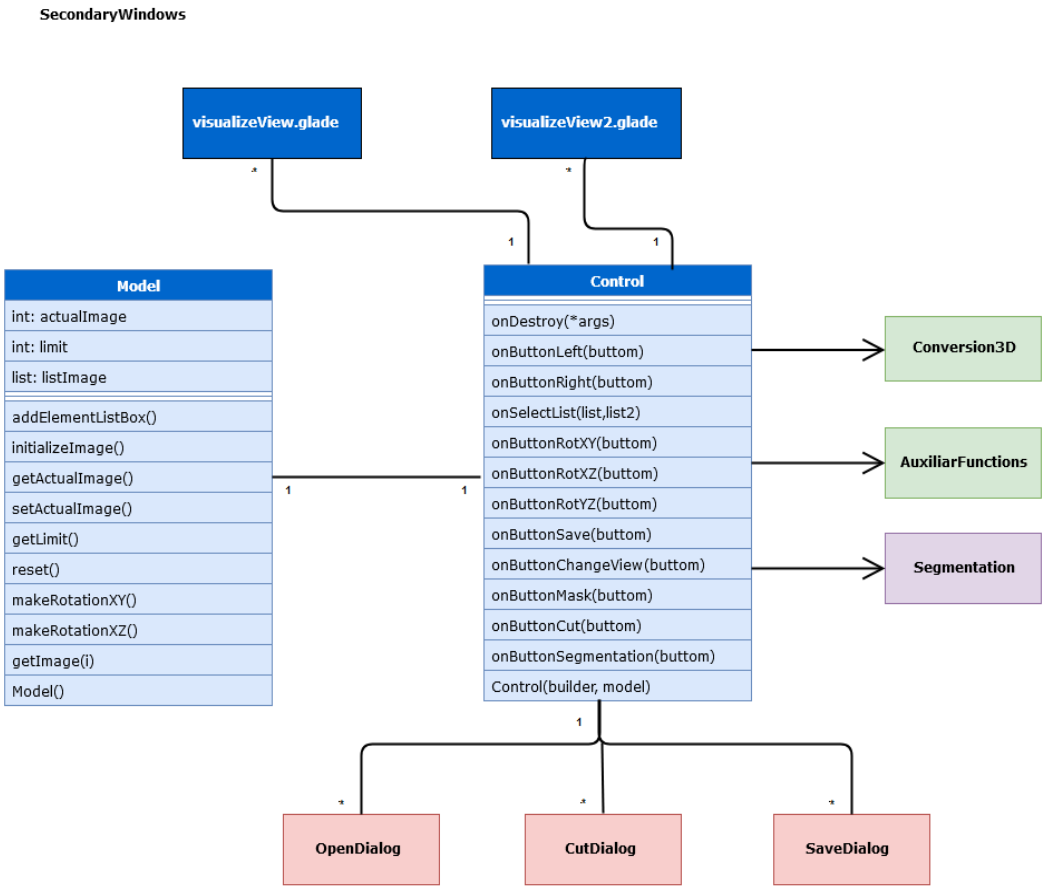


Figura 7.8: **Diagrama de clases de la ventana visualización.**

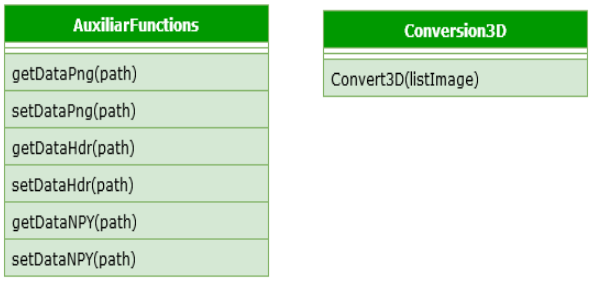


Figura 7.9: **Diagrama de clases de las funciones auxiliares.**

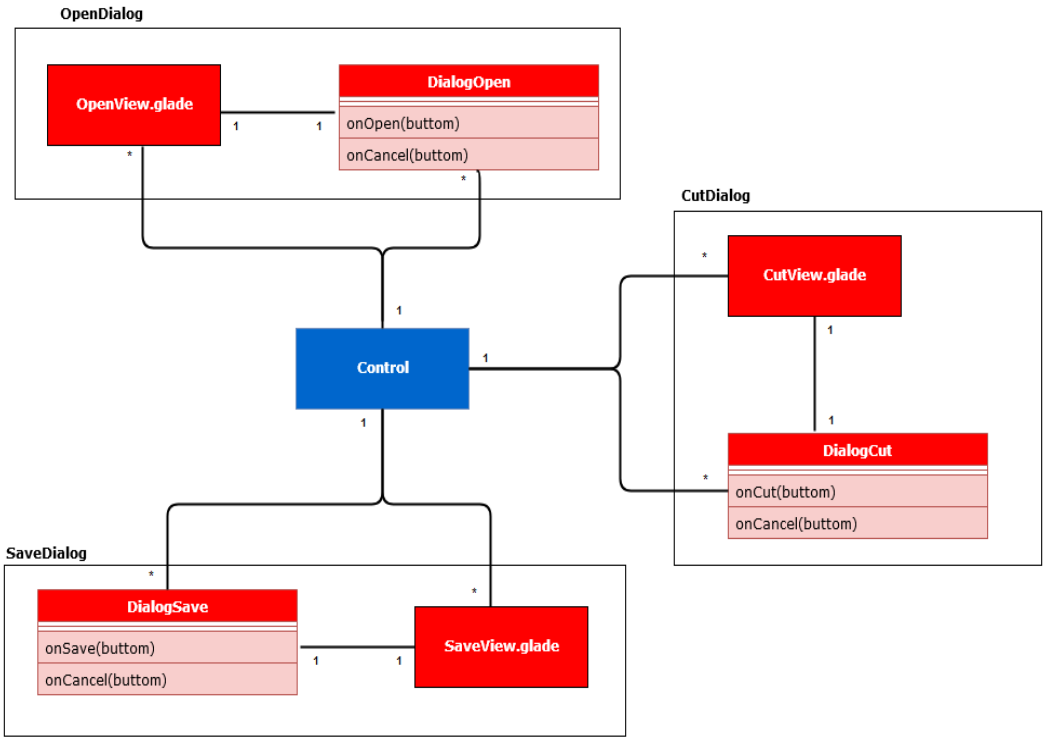


Figura 7.10: **Diagrama de clases de las ventanas diálogos.**

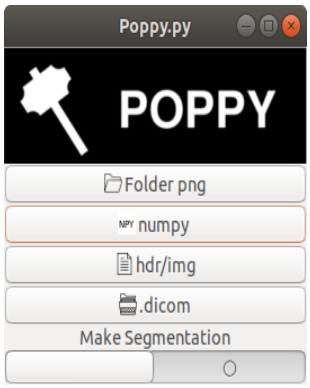


Figura 7.11: **Pantalla principal**

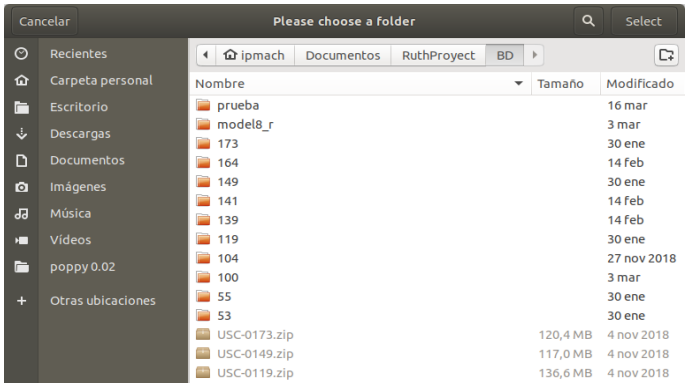


Figura 7.12: **Pantalla de selección de archivos**

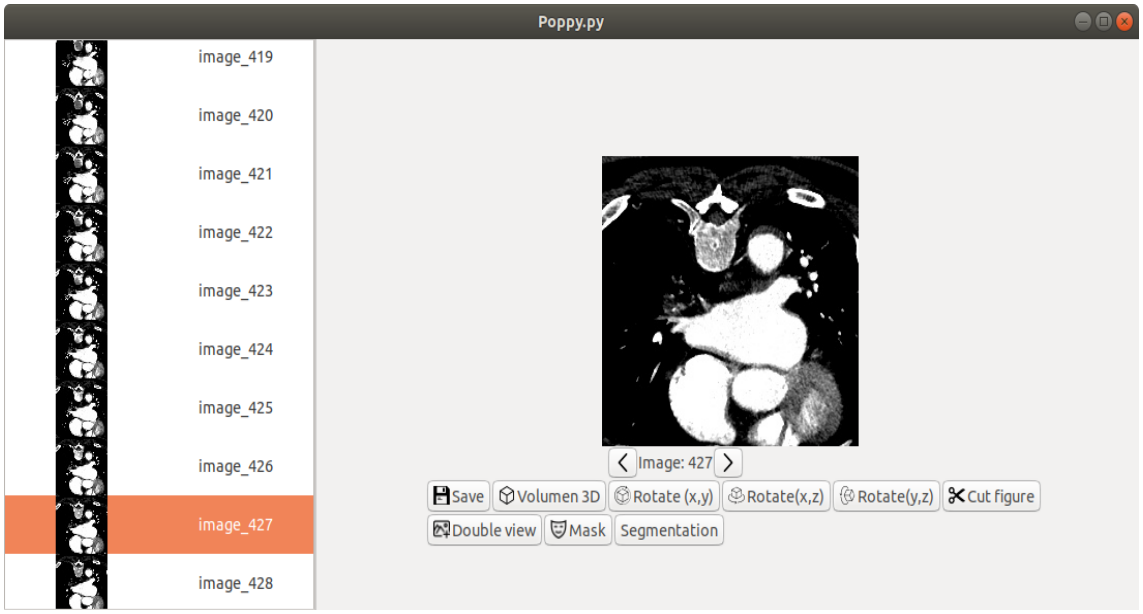


Figura 7.13: **Pantalla de visualización individual de imágenes**

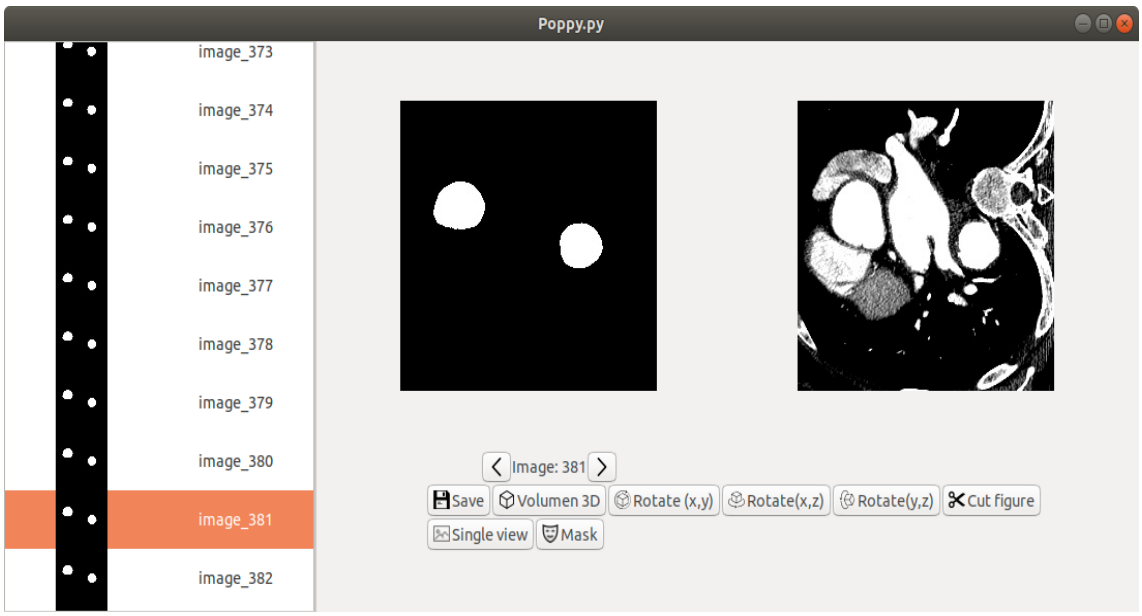


Figura 7.14: **Pantalla de visualización doble de imágenes y resultados de la segmentación**

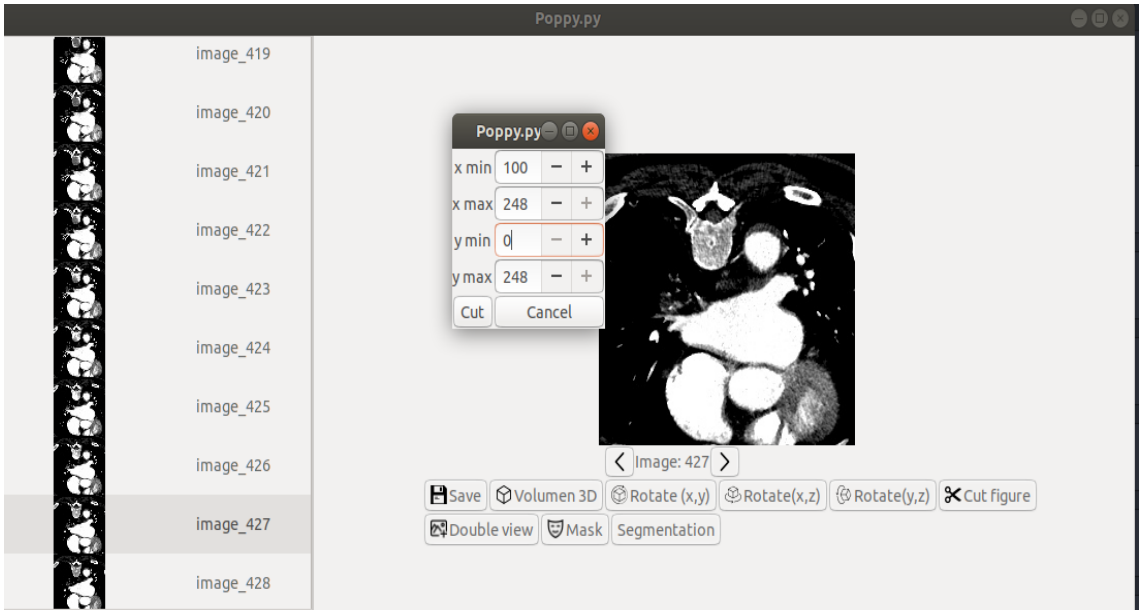


Figura 7.15: **Función para cortar figura**

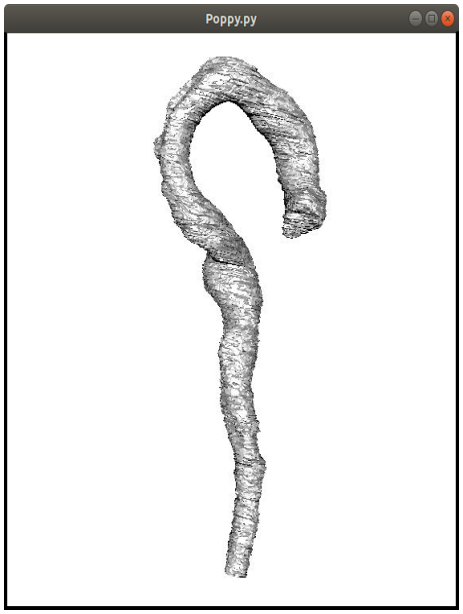


Figura 7.16: **Visualización de volumen 3D**

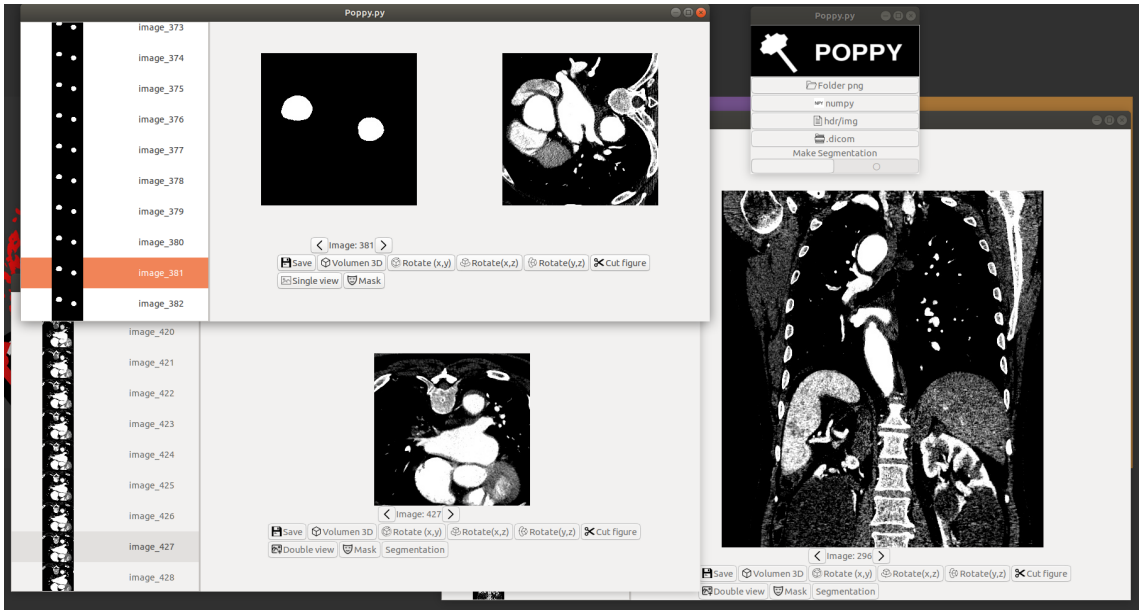


Figura 7.17: **Captura de pantalla del visualizador**

Tabla 7.8: **Pruebas de caja negra**

Código prueba	Descripción	Resultado
E001	Al abrir un nuevo fichero seleccionas el formato incorrecto.	EC
E002	Al segmentar seleccionas un fichero con el tamaño incorrecto.	EC
E003	Abrir un fichero correctamente.	FC
E004	Abrir un fichero corrupto.	EC
E005	Realizar una segmentación correcta.	FC
S001	Guardar un fichero en un formato distinto.	FC
M001	Cortar una figura correctamente.	FC
M002	Intentar cortar figura saliéndote del tamaño de la figura.	NP
M003	Rotar figura.	FC
M004	Aplicar máscara mismas proporciones.	FC
M005	Aplicar máscara de distintas proporciones.	EC
M006	Realizar cambios de simple a doble correctamente.	FC
M007	Realizar cambios de simple a doble de distintas proporciones.	EC
M008	Realizar cambios de doble a simple correctamente.	FC

[32] y *PIP* [33]. Los comandos usados para realizar las instalaciones de librerías requeridas para la ejecución de la aplicación son los siguientes:

- *conda install numpy*
- *conda install tensorflow*
- *conda install keras*
- *conda install opencv-python*
- *conda install scikit-image*
- *pip/pip3 install medpy*
- *pip/pip3 install PyGObject* [1]

Todas las dependencias anteriores se podían instalar con *PIP* sin embargo se recomienda usar *miniconda* para mejorar su eficiencia. Esto es debido a que la compilación a nivel ensamblador es mejor en los módulos que no están escritos en python. Para ejecutar el programa entre en el directorio de la aplicación y ejecute el siguiente comando *python main.py* o *python3 main.py*, el cual abrirá el menú principal.
En cuanto a los requisitos hardware necesarios para ejecutar la aplicación, se necesita como mínimo un procesador *Intel Core 3* y 4GB de memoría *RAM*, siendo óptimo disponer de un procesador *Intel Core 5* con 8 GB de *RAM* y una tarjeta gráfica con soporte para *deep learning* como por ejemplo una *NVIDIA 960 GTX*.

[1] Este comando no funciona en windows donde se deben seguir otros pasos [34].

Capítulo 8

Conclusiones

La arteria aorta es el principal vaso del cuerpo humano, siendo su función principal el aporte de oxígeno a los órganos vitales que lo conforman. Su cuidado es pues fundamental, dado que un mal funcionamiento puede desencadenar graves consecuencias para la vida del paciente. La gran mayoría de enfermedades de la aorta pueden detectarse tanto por su forma como por la de su lumen. En ambos casos, el uso de imágenes de tomografía computarizada (CT) facilita su diagnóstico. Sin embargo, a la hora de valorar de forma más exhaustiva el estado de la aorta, realizar un análisis cuantitativo detallado resulta del todo fundamental. Para ello es necesario proceder previamente a una segmentación de la aorta, operación que puede resultar compleja, debido a la presencia de otro tipo de órganos adyacentes de similar radiopacidad.

El principal objetivo de este proyecto fue la creación de un algoritmo capaz de segmentar la aorta mediante el uso de métodos de aprendizaje automático. Para ello se apostó por el uso de técnicas de segmentación basadas en redes neuronales convolucionales. La principal ventaja de estas técnicas es su capacidad para adaptarse a formas complejas, siendo además posible la mejora de los resultados obtenidos a medida que se incorporan nuevos datos para su entrenamiento. Para la gestión de los resultados obtenidos, se ha desarrollado una aplicación de escritorio, que facilita su visualización y que además, puede facilitar en el futuro la mejora del algoritmo, lo cual resulta una ventaja añadida nada desdeñable.

Respecto a la arquitectura de redes utilizada, se hizo uso de la llamada *UNet*. Esta arquitectura ha sido específicamente diseñada para su uso en segmentación de imagen médica. Su principal característica es su capacidad para poder obtener resultados óptimos, sin la necesidad de utilizar grandes bases de datos para su entrenamiento. En nuestro caso, hemos utilizado 24 capas lo que ha supuesto el ajuste de 2.624.020 parámetros. Para realizar dicho ajuste, solamente han sido necesarias 3.172 imágenes de las cuales 2220 fueron usadas para el entrenamiento,

siendo destacable el hecho de que no se haya utilizado *data augmentation* para aumentar el número de casos de la base de datos de entrenamiento, ni modelos pre-entrenados. Así mismo, reseñar que los datos de entrada de la red fueron filtrados inicialmente por un algoritmo de preprocesado consistente en un filtro que simplifica la información recibida, eliminando parte del ruido. Los datos de salida de la red fueron posteriormente procesados por otro algoritmo que realiza un seguimiento de la estructura de la aorta.

Para la evaluación final de los resultados obtenidos en el proceso de segmentación, se ha utilizado el *Coeficiente Sørensen–Dice*. Los valores obtenidos para la segmentación de la aorta completa fueron desde 0,76 para el peor de los casos, hasta 0,92 para el mejor, con una media de 0,86. Por lo que respecta a la parte correspondiente a la aorta descendente hasta el arco aórtico, los resultados mejoraron como cabría esperar al no tener que segmentar la entrada al corazón, que resultó ser la parte más compleja. Los resultados alcanzados en este caso fueron desde 0,84 (peor de los casos) hasta 0.92 con una media de 0.87.

Para la gestión de los resultados obtenidos se ha desarrollado una aplicación gráfica de escritorio, formada por ventanas, para la visualización de los distintos cortes de los casos. Es de destacar la inclusión de un menú para la lectura de casos en distintos formatos, tales como *png*, *numpy*, *analyze* y *dicom*. La aplicación permite además la rotación de la imagen en los diferentes ejes, el recorte para la extracción de regiones de interés 3D en el volumen de la imagen, la inclusión de máscaras para la comparación de resultados y la posibilidad de guardar resultados en diferentes formatos de imagen (*png*, *numpy* y *analyze*). La aplicación permite además visualizar los cortes de forma individual, comparar dos imágenes y realizar la reconstrucción 3D de la aorta a partir de los resultados obtenidos en la segmentación.

Como conclusión final, podemos afirmar que se han cumplido todos los objetivos propuestos inicialmente en el proyecto. La parte más complicada de desarrollar fue la relativa a la segmentación debido a la presencia de múltiples estructuras y órganos con-radiopacidades similares que dificultan la tarea.

Respecto al trabajo futuro se proponen las siguientes mejoras:

- Realización pruebas de usabilidad de la aplicación con usuarios reales.
- Optimización en el manejo de datos por parte de la aplicación, reducción de tiempos de carga y resolución de problemas de conversión entre formatos.
- Aumento de las funcionalidades de la aplicación.

- Mejora del algoritmo de segmentación que permita la extracción integral de la aorta en un solo paso.

Apéndices

Apéndice A

Arteria aorta

La aorta es la arteria más grande del cuerpo. La aorta comienza en la parte superior del ventrículo izquierdo, la cámara de bombeo muscular del corazón. El corazón bombea sangre del ventrículo izquierdo a la aorta a través de la válvula aórtica. La aorta se divide en cuatro secciones (Figura A.1) [35]:

- **Aorta ascendente**: se eleva desde el corazón y mide aproximadamente 2 pulgadas de largo.
- **Arco aórtico**: parte curva de la aorta que une la ascendente y la torácica. Proporciona sangre a los brazos, cuello y la cabeza.
- **Aorta torácica descendente**: baja a través del pecho. Sus pequeñas ramas suministran sangre a las costillas y a algunas estructuras del pecho.
- **Aorta abdominal**: comienza en el diafragma, dividiéndose para convertirse en las arterias ilíacas emparejadas en la parte inferior del abdomen. La mayoría de los órganos principales reciben sangre de las ramas de la aorta abdominal.

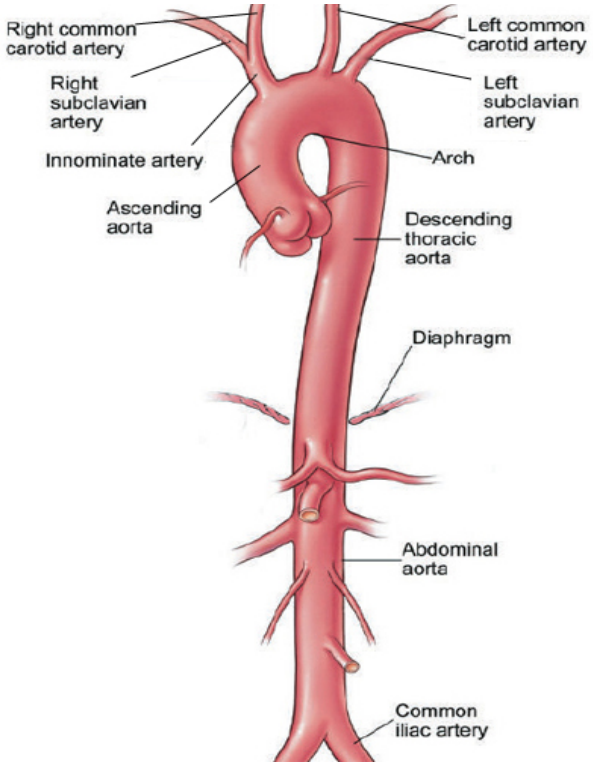


Figura A.1: **Partes de la aorta.** [1]

Apéndice B

Escala Hounsfield

La escala de Hounsfield es una unidad estándar en imágenes de tomografía computorizada (Figura B.1). Se calcula a partir de una transformación lineal de las medidas del coeficiente de atenuación. Dicho coeficiente mide la facilidad con que los rayos X penetran el material. Esta transformación convierte los valores originales en los valores estimados con la radio densidad del agua destilada y el aire a temperatura estándar (STP).

$$HU = 1000 \times \frac{\mu - \mu_{agua}}{\mu_{agua} - \mu_{aire}} \tag{B.1}$$

donde μ es el coeficiente de atenuación lineal de entrada, μ_{agua} el coeficiente de atenuación del agua y μ_{aire} el coeficiente de atenuación del aire.

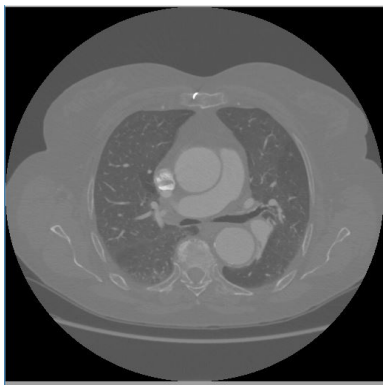


Figura B.1: **Imagen en escala Hounsfield.**

Apéndice C

Generadores

Los generadores son funciones o clases que nos permiten no tener cargado en memoria toda la base de datos en el proceso de entrenamiento. Un generador se ocupa de cargar solo del tamaño del *batch size* necesario y que los datos no se repitan en el proceso. Para este proyecto se creo un generador heredando la propia clase disponible en *keras.utils.Sequence* [36] (Código C.1) . Al usar una versión modificada del propio generador de *Keras* permite la paralelización de los distintos *batch* en el mismo *epoch* aumentado así la velocidad de entrenamiento.

Código C.1: Generador usado en el proyecto

```
#Class generator that inherit from keras.utils.Sequence
#Worls like a interface to work with the tensorflow library to work with a
    generator in different threads
class Data_generator(keras.utils.Sequence):
    #To initialice the object
    #x_set: index of all the files x in the BD [i for i in range(len(BD))]
    #y_set: the same of x_set
    #batch_size: batch_size use for the training process.
    def __init__(self, x_set, y_set, batch_size,file_name_x, file_name_y):
        self.x, self.y = x_set, y_set
        self.batch_size = batch_size
        self.file_name_x = file_name_x
        self.file_name_y = file_name_y
    #Give the number of batch we can take from the DB
    def __len__(self):
        return self.x.shape[0] / self.batch_size
    #get a batch of x,y
    #idx: number of the batch
    def __getitem__(self, idx):
        batch_x = self.x[idx * self.batch_size:(idx + 1) *
        self.batch_size]
        batch_y = self.y[idx * self.batch_size:(idx + 1) *
        self.batch_size]
```

```
23          x = np.take(np.load(self.file_name_x),batch_x, axis =0)
24          y = np.take(np.load(self.file_name_y), batch_y, axis = 0)
25          return x.reshape((x.shape[0],x.shape[1],x.shape[2],1)),y.reshape([y.
    shape[0],y.shape[1],y.shape[2],1])
```

Apéndice D

Operaciones morfológicas

Las operaciones morfológicas son fáciles de usar y funcionan sobre la base de la teoría de conjuntos. El objetivo de utilizar la información morfológica en operaciones es eliminar las imperfecciones en la estructura de una imagen. La mayoría de las operaciones se basan en la combinación de dos procesos: dilatación y erosión. Estas operaciones utilizan una pequeña matriz llamada elemento estructural que funciona como un filtro convolución pero solo para imágenes binarias [37].

La **dilatación** de A por el elemento estructural B viene definida por:

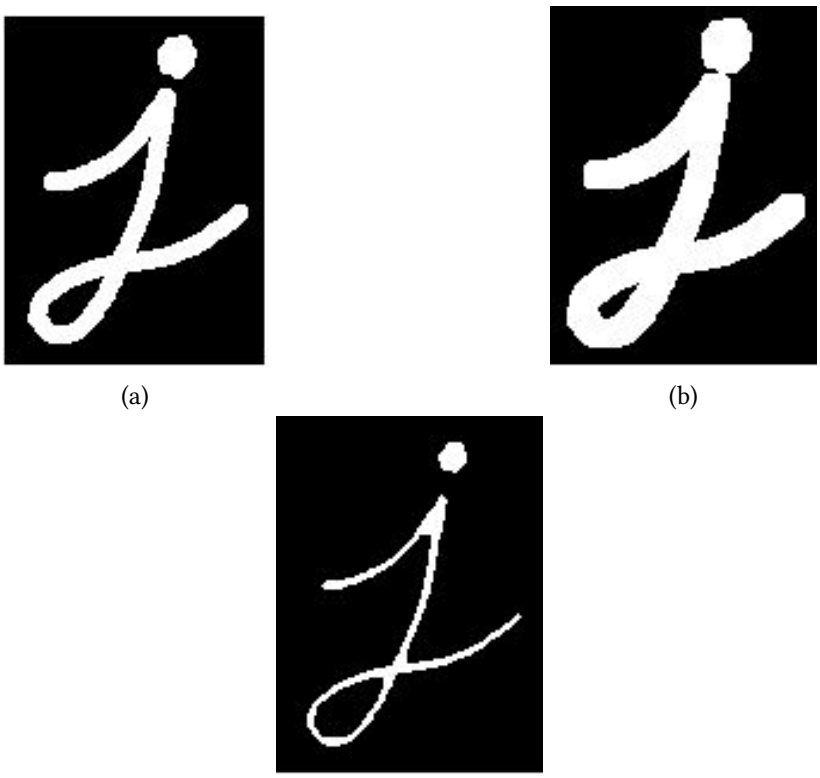
$$A \oplus B = \bigcup_{b \in B} A_b \tag{D.1}$$

donde su función es aumentar las estructuras que están dentro de la imagen A a partir de la matriz B (Figura D.1b).

La **erosión** de A por el elemento estructural B viene definida por:

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{D.2}$$

donde su función es reducir las estructuras que están dentro de la imagen A a partir de la matriz B (Figura D.1c).



(a) (b)

(c)

Figura D.1: **Operaciones morfológicas**: a) Imagen original. b) Imagen dilatada. c) Imagen erosionada. Imágenes sacadas de la guía de operaciones morfológicas para OpenCV [2].

Bibliografía

[1] C. Clinic, "Imagen de la aorta." software available from my.clevelandclinic.org/. [Online]. Available: https://my.clevelandclinic.org/health/articles/17058-aorta-anatomy

[2] "Tutorial morphological operations opencv," software available from docs.opencv.org. [Online]. Available: https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

[3] M. Alemán-Flores, D. Santana-Cedrés, L. Alvarez, A. Trujillo-Pino, L. Deniz, P. Tahoces, and J. M. Carreira, *Segmentation of the Aorta Using Active Contours with Histogram-Based Descriptors: 7th Joint International Workshop, CVII-STENT 2018 and Third International Workshop, LABELS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings*, 09 2018, pp. 28–35.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[5] C. D. Fellbaum, "Imagenet proyect," 2010–, software available from image-net.org. [Online]. Available: http://www.image-net.org/

[6] J.-Q. Zheng, Q.-B. L. Xiao-Yun Zhou, and G.-Z. Y. Celia Riga, *Abdominal Aortic Aneurysm Segmentation with a Small Number of Training Subjects*, 04 2018. [Online]. Available: https://arxiv.org/abs/1804.02943

[7] K. Román, N. Aranjuelo, L. Kabongo, G. Maclair, N. Lete, M. Ceresa, A. García-Familiar, I. Macía, and M. González Ballester, "Fully automatic detection and segmentation of abdominal aortic thrombus in post-operative cta images using deep convolutional neural networks," *Medical Image Analysis*, vol. 46, 03 2018.

[8] L. Alvarez, E. González, C. Cuenca, A. Trujillo, P. G. Tahoces, and J. M. Carreira, "Ellipse motion estimation using parametric snakes," *Journal of Mathematical*

Imaging and Vision, vol. 60, no. 7, pp. 1095–1110, Sep 2018. [Online]. Available: https://doi.org/10.1007/s10851-018-0798-9

[9] M. Nielsen, "Neural networks and deep learning," 2015, http://neuralnetworksanddeeplearning.com/.

[10] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." MIT Press, 2016, pp. 298–301, http://www.deeplearningbook.org.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[13] E. B.-N. Maria Elena Rodriguez-Salazar, Sergio alvarez-Hernandez, "Coeficientes de asociacion." Plaza y Valdes, 2001.

[14] G. van Rossum, "Python," 1990–, software available from python.org. [Online]. Available: https://www.python.org/

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[16] F. Chollet *et al.*, "Keras," 2015, software available from keras.io. [Online]. Available: https://keras.io/

[17] T. Oliphant, "NumPy: A guide to NumPy," 2006–, software available from numpy.org. [Online]. Available: http://www.numpy.org/

[18] C. Analytics, "Numba: A guide to Numba," 2012–, software available from numba.pydata.org. [Online]. Available: https://numba.pydata.org/

[19] S. Kimball and P. Mattis, "Gtk," 1998–, software available from gtk.org. [Online]. Available: https://www.gtk.org/

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[21] "Medpy," 2013–, software available from loli.github.io. [Online]. Available: https://loli.github.io/medpy/generated/medpy.io.load.load.html

[22] NVIDIA, "Cuda in deep learning frameworks." 2007–, software available from developer.nvidia.com/. [Online]. Available: https://developer.nvidia.com/deep-learning-frameworks

[23] "Google colab," software available from colab.research.google.com. [Online]. Available: https://colab.research.google.com/

[24] "Jupyter notebooks," software available from jupyter.org. [Online]. Available: https://jupyter.org/

[25] "Analyze 7.5 file format," software available from rportal.mayo.edu/bir/ANALYZE75.pdf. [Online]. Available: https://rportal.mayo.edu/bir/ANALYZE75.pdf

[26] L. Lamport, "Latex," 1983–, software available from latex-project.org. [Online]. Available: https://www.latex-project.org/

[27] Google, "Google drive," 2012–, software available from google. [Online]. Available: https://www.google.com/drive/

[28] J. Hamano, "Git," 2005–, software available from git-scm.com. [Online]. Available: https://git-scm.com/

[29] "draw.io," software available from draw.io. [Online]. Available: https://about.draw.io/about-us/

[30] H. Takeuchi and I. Nonaka, "Scrum," 1986 –, software available from www.mountaingoatsoftware.com. [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum

[31] I. Jolliffe, *Principal Component Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_455

[32] I. Anaconda, "Anaconda," 2012 –, software available from www.anaconda.com. [Online]. Available: https://www.anaconda.com/

[33] C. por la comunidad, "Pip," 2011 –, software available from pypi.org. [Online]. Available: https://pypi.org/project/pip/

[34] "Install pyobject," software available from pygobject.readthedocs.io. [Online]. Available: https://pygobject.readthedocs.io/en/latest/getting_started.html

[35] MedicineNet, "Definición y anatomía de la aorta." software available from www.medicinenet.com. [Online]. Available: https://www.medicinenet.com/image-collection/aorta_picture/picture.htm

[36] F. Chollet *et al.*, "Keras utils," 2015, software available from github. [Online]. Available: https://github.com/keras-team/keras/blob/master/keras/utils/data_utils.py#L305

[37] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. [Online]. Available: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X